
# Workshop #19: MSDM 2009

# Multi-Agent Sequential Decision-Making in Uncertain Domains

## 4th Workshop in the MSDM Series

Held in conjunction with AAMAS-2009 (the 8th International Joint Conference on Autonomous Agents and Multiagent Systems), in Budapest, Hungary on May 11th, 2009.

## Organizing Committee/Editors


| | |
|---|---|
| Christopher Amato | University of Massachusetts, Amherst |
| Janusz Marecki | IBM T.J. Watson Research Center |
| Sven Seuken | Harvard University |
| Matthijs Spaan | Instituto Superior Técnico |

## Program Committee

| | |
|---|---|
| Martin Allen | Connecticut College |
| Daniel Bernstein | University of Massachusetts, Amherst |
| Aurelie Beynier | University Pierre and Marie Curie (Paris 6) |
| François Charpillet | LORIA |
| Dmitri Dolgov | Stanford University |
| Prashant Doshi | University of Georgia |
| Ed Durfee | University of Michigan |
| Alberto Finzi | University of Roma |
| Piotr Gmytrasiewicz | University of Illinois Chicago |
| Robert Goldman | Smart Information Flow Technologies |
| Sven Koenig | University of Southern California |
| Rajiv Maheswaran | University of Southern California |
| Abdel-Illah Mouaddib | Universit de Caen |
| David Musliner | Smart Information Flow Technologies |
| Frans Oliehoek | University of Amsterdam |
| Praveen Paruchuri | University of Southern California |
| John Phelps | Smart Information Flow Technologies |
| David Pynadath | Information Sciences Institute |
| Zinovi Rabinovich | University of Southampton |
| Anita Raja | University of North Carolina at Charlotte |
| Jeffrey Rosenschein | Hebrew University |
| Jiaying Shen | SRI International, Inc. |
| Milind Tambe | University of Southern California |
| Ping Xuan | Clark University |
| Shlomo Zilberstein | University of Massachusetts, Amherst |

# Table of Contents

# Quadratic Programming for Multi-Target Tracking

Raghav Aras
LORIA
Campus Scientifique
Vandeouvre-lès-Nancy,
FRANCE
raghav.aras@gmail.com

Alain Dutech
INRIA
Campus Scientifique
Vandeouvre-lès-Nancy
FRANCE
dutech@loria.fr

François Charpillet
INRIA
Campus Scientifique
Vandeouvre-lès-Nancy,
FRANCE
charpillet@loria.fr

## ABSTRACT

We consider the problem of tracking multiple, partially observed targets using multiple sensors arranged in a given configuration. We model the problem as a special case of a (finite horizon) DEC-POMDP. We present a quadratic program whose globally optimal solution yields an optimal tracking joint policy, one that maximizes the expected targets detected over the given horizon. However, a globally optimal solution to the QP cannot always be found since the QP is nonconvex. To remedy this, we present two linearizations of the QP to equivalent 0-1 mixed integer linear programs (MIPs) whose optimal solutions, which may be always found through the branch and bound method, for example, yield optimal joint policies. Computational experience on different sensor configurations shows that finding an optimal joint policy by solving the proposed MIPs is much faster than using existing algorithms for the problem.

## 1. INTRODUCTION

This paper addresses a special case of finite horizon DEC-POMDPs. The special case has been called a network distributed POMDP [4] or a factored DEC-POMDP [5]. Lately, this special case has received attention in these pages, especially for the problem of detecting multiple targets passing through a given configuration of locations using multiple sensors, and specialized algorithms have been conceived for it [4], [6], [3]. Our focus too shall be on the multi-target tracking problem.

In this problem, the set of agents (sensors) is partitioned into subsets. It is assumed that for each subset, we can define immediate rewards that are dependent on the actions of the agents of the subset but not on the actions of agents outside the subset. It is furthermore assumed that the probabilities with which an agent receives observations are independent of probabilities with which other agents receive observations. Finally, it is assumed that the probabilities of transitions between states are independent of actions of the agents.

The purpose of the above partitioning scheme is to model autonomy for agents in one subset from those in other subsets. In the multi-target tracking problem (Figure 1), only the two sensors surrounding a location are required to detect a target at that location; the other sensors play no role in the target's detection. Each sensor enjoys autonomy from all the other sensors save its immediate neighbor(s) in any of the four cardinal directions. In this problem, state transition probabilities are independent of the sensors' actions since a sensor's choice of location to monitor (its action) does not influence the targets' positions (the state of the problem).

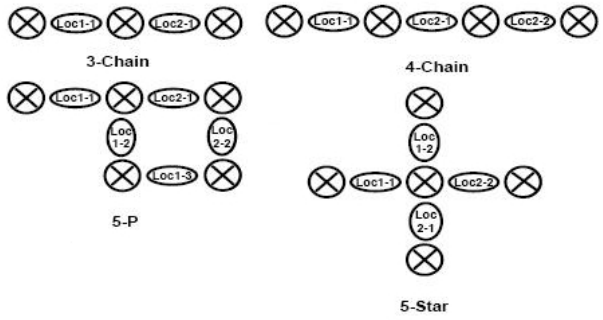


**Figure 1: Sensor configurations (reprised from [6]).**

Recently, in [1], mathematical programming was applied with encouraging results to the general case of finite horizon DEC-POMDPs. Specifically, a nonlinear program (nonlinear objective, linear constraints) and 0-1 mixed integer linear programs (MIPs), equivalent to the nonlinear program, were presented. The use of the sequence form of a policy facilitated the conception of these programs. Exploiting the power of ILOG CPLEX and NEOS solvers, these programs were able to solve sample problems much more rapidly than other algorithms (by two orders of magnitude).

In this paper, we adapt this mathematical programming approach to the multi-target tracking problem. For the configurations of locations considered, the set of agents partitions into subsets that each contain exactly two agents. This being so, the adaptation of the nonlinear program yields a quadratic program. Following [1], we linearize this QP to a 0-1 MIP. While solving the QP is only guaranteed to find a locally optimal joint policy (but in practice often finds the optimal joint policy), the solving MIP is guaranteed to return an optimal joint policy. Secondly, we present a *new* linearization of the quadratic program to a 0-1 MIP. The new 0-1 MIP uses exponentially fewer variables and constraints than the other 0-1 MIP. This new MIP is in fact also usable for solving the general case of DEC-POMDPs. Computational experience of the programs on different location configurations reveals that the programs are much faster than existing approaches for the problem, the improvement

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

in time being of the same order as for problems of general, unpartitioned DEC-POMDPs.

## 2. THE MODEL

The special case of a DEC-POMDP is specified by the following data:

$I = \{1, 2, \ldots, n\}$, a set of agents. $S$, a set of states. $A_i$ and $O_i$, sets of respectively actions and observations of agent $i \in I$. For each pair of states $s, s' \in S$, the probability $p(s, s')$ that the process moves from $s$ to $s'$, is defined. As stated earlier, we assume that this probability is not conditional on the actions of the agents. For each $i \in I$, for each $a \in A_i$, for each $s' \in S$, and for each $o \in O_i$, the probability $q_i(a_i, o_i, s')$ that $i$ receives $o$ in $s'$ if he has taken $a$, is defined. Again, as stated earlier, we assume that this probability is not conditional on the observations or actions of other agents.

The set $I$ of agents is partitioned into subsets which together exhaust it. The set of these subsets is denoted by $D$. Each subset in $D$ has one or more agents in common with at least one other subset in $D$.

For each $d \in D$, let $A^d$ denote the set $\times_{i \in d} A_i$ (joint actions over $d$). Thereby, for each $d \in D$, immediate rewards are defined: that is, for every $s \in S$ and for every joint action $d \in A^d$, the reward $R^d(s, a)$ of the agents of $d$ taking $a$ in $s$, is defined. Thereby, the total reward obtained by the agents in a period if the state is $s$ and they take the joint action $a \in A$ is $\sum_{d \in D} R^d(s, a(d))$ where $a(d) \in A^d$ is the joint action over $d$ formed by the elements of $a$.

Let $T$ denote the horizon of the problem. Let $Z_i$ denote the set of all possible sequences of $T$ - 1 or less observations conceivable from $O_i$. $Z_i$ also includes the null sequence. The *policy* of an agent is a function from $Z_i$ to $A_i$. In using a policy $\pi_i$, the agent takes action $\pi_i(z)$ in a period if the sequence of observations he has received till that period is $z$. A *mixed* policy is a probability distribution over the set of policies. A *joint policy* is a $n$-tuple of policies, one policy in the tuple per agent.

Let $\Delta(S)$ denote the set of probability distributions over $S$. For $b^* \in \Delta(S)$, an *optimal joint policy* at $b^*$ is a joint policy $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ that maximizes the expectation of $\sum_{t=1}^{T} \sum_{d \in D} R^d(s^t, \pi^d(z^{d,t}))$ where $s^t$ is a random variable representing the state in period $t$, $z^{d,t}$ is a random variable representing the tuple of sequences of observations received by the agents in $d$ till period $t$, and $\pi^d(z^{d,t})$, the corresponding joint action according to $\pi^d$, the joint policy over $d$ formed from $\pi$, and where $s^1$ is according to $b^*$.

### 2.1 Example

We follow the specifications of the multi-target tracking problem given in [4]. Consider the 4-chain configuration given in Figure 1. The four sensors in the chain together are meant to detect mobile objects (targets) that appear at the three locations following a fixed Markovian law. Two types of targets are possible: targets of type 1 appear only at location Loc1-1 while targets of type 2 can appear at locations Loc2-1 or Loc2-2.

Each sensor is an agent. Sensors 1 and 2 are assigned to monitor location Loc1-1, sensors 2 and 3 are assigned to monitor location Loc2-1 while sensors 3 and 4 are assigned to monitor location Loc2-2. Thus, $I = \{1, 2, 3, 4\}$ and $D = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$.

In a period, a sensor can either monitor the location to its left ($\supset$) or monitor the location to its right ($\subset$) or switch itself off ($\varnothing$). A target is detected at a location only if the location is being monitored by the sensors to its left and right (or, in other configurations such as Five-P, by the sensors to its top and bottom); if only one of these two sensors is monitoring the location, the target slips by undetected. For instance, if a target is at location Loc1-1 in a period, it is detected only if sensor 1 chooses $\subset$ and sensor 2 chooses $\supset$, and if a target is at location Loc2-2 in a period, it is detected only if sensor 2 chooses $\subset$ and sensor 3 chooses $\supset$. If targets appear at these two locations in the same period, sensor 2 must decide which location is preferable to monitor in that period.

The state of the problem in a period is described by the a pair of numbers $(x, y)$ with $x \in \{0, 1\}$ and $y \in \{0, 1, 2\}$. $x = 0$ denotes the absence of target 1 in Loc1-1 and $x = 1$ its presence in Loc1-1. $y = 0$ denotes the absence of target 2 in both Loc2-1 and Loc2-2, $y = 1$ its presence in Loc2-1 and $y = 2$ its presence in Loc2-2. There are thus 6 possible states in the problem. The state of the problem evolves from one period to another according to probabilities that are independent of the agents' actions.

A target arriving at a location is partially observed by the sensors assigned to the location and unobserved by the sensors not assigned to the location. The observations received by a sensor assigned to a location are '1' (for presence of a target) and '0' (for absence of target).

Finally, each monitoring action of a sensor in a period obtains a reward. The monitoring of a target-less location has a small negative reward. The detection of a target at a location results in a large positive reward obtained collectively by *all* the sensors. If a sensor switches itself off, there is no reward. Our objective in this problem is to maximize the expected number of targets detected in a given number of periods, and to do so we must maximize the total expected reward obtainable in those periods.

## 3. THE SEQUENCE FORM

The *sequence form* of a policy is a representation of a (possibly, mixed) policy that facilitates formulating the problem of finding an optimal joint policy as a mathematical program. It was introduced in [2] for games in extensive form and used in [1] for finite horizon DEC-POMDPs.

The sequence form of a policy of an agent is defined as a conditional probability distribution over the set of histories of the agent. We define a history of length $t \geq 1$ of an agent to be a sequence of length $2t$ - 1 in which the elements in odd positions are actions of the agent and those in even position, his observations. Thus, a history of length $t$ has $t$ actions and $t$ - 1 observations. A history of length 1 is just an action. A history of length $T$ shall be called *terminal*.

To give examples of histories from the 3 or 4-chain configuration of the multi-target tracking problem where $A_i = \{\subset, \supset, \varnothing\}$ and $O_i = \{0, 1\}$, $\subset$-1-$\supset$-0-$\subset$ is a history of length 3, $\supset$-0-$\supset$ is a history of length 2, $\varnothing$ is a history of length 1 etc.

Let $W_i$ denote the set of histories of lengths less than or equal to $T$ of agent $i$. Let $Y_i$ denote the set of terminal histories of agent $i$. A policy in the sequence form of agent

$i$ is a function $\sigma_i$ from $W_i$ to [0, 1] such that,

$$\sum_{a \in A_i} \sigma_i(a_i) = 1 \quad (1)$$

$$\sigma_i(h) = \sum_{a \in A_i} \sigma_i(h, o, a), \quad \forall h \in W_i \backslash Y_i, \forall o \in O \quad (2)$$

where $h, o, a$ denotes the history obtained on concatenating $o$ and $a$ to $h$. For history $h = (a^1, o^1, a^2, \ldots, o^{t-1}, a^t)$ $\sigma_i(h)$ is the conditional probability,

$$\text{prob}(a^1, a^2, \ldots, a^t | o^1, o^2, \ldots, o^{t-1}) \quad (3)$$

In using a policy $\sigma_i$ in the sequence form, the agent takes action $a$ in period $t$ upon receiving observation $o$ in that period with probability $\sigma_i(h_t, o, a)/\sigma_i(h_t)$, where $h_t$ is the history that has occurred till that period.

(Note: That any (mixed) policy $\pi_i$ in the canonical form can be converted to its equivalent sequence form is self-evident. The converse can also be shown: For every policy $\sigma_i$ in the sequence form, there exists a (possibly, mixed) policy $\pi_i$ in the canonical form such that for each history $h \in W_i$, the conditional probability of the form (3) assigned to $h$ by $\sigma_i$ is the same as assigned by $\pi_i$.)

## 3.1 The Expected Reward

The expected reward of a joint policy (in the sequence form) can be expressed in terms of the sum of the expected rewards of the terminal joint histories of each subset $d \in D$. Denoting the size of $d$ by $m$, a terminal joint history of $d$ is an $m$-tuple of terminal histories, one history in the tuple per agent in $d$.

Let $Y^d$ denote the set $\times_{i \in d} Y_i$ of of terminal joint histories of $d$. In a joint history $J \in Y^d$, let $J_i$ denote the history of agent $i \in d$ in $h$. Let $r^d(J)$ denote the expected reward of $J \in Y^d$. Then, the expected reward of a joint policy ($\sigma_1$, $\sigma_2$, …, $\sigma_n$) in the sequence form is given by,

$$\sum_{d \in D} \sum_{J \in Y^d} r^d(J) \prod_{i \in d} \sigma_i(J_i) \quad (4)$$

Let $O^d$ denote the set $\times_{i \in d} A_i$ (joint observations over $d$). For a terminal joint history $J = (a^1, o^1, a^2, o^2, \ldots, o^{T-1}, a^T) \in Y^d$ , where each $a^k$ is a joint action in $A^d$ and each $o^k$ is a joint observation in $O^d$. Thereby,

$$r^d(J) = \{ \prod_{t=1}^{T-1} P(o^t | b^{d,t}, a^t) \} \{ \sum_{t=1}^{T} R^d(b^{d,t}, a^t) \}$$

The elements appearing in the r.h.s. are as follows. For $t = 1$, $b^{d,t} = b^*$; for $t > 1$, for each $s' \in S$,

$$b^{d,t}(s') = \sum_{s \in S} b^{d,t-1}(s) p(s, s') \prod_{i \in d} q_i(a_i^{t-1}, s', o_i^{t-1})$$

, for each $t \geq 1$,

$$R^d(b^{d,t}, a^t) = \sum_{s \in S} b^{d,t}(s) R^d(s, a^t)$$

and,

$$P(o^t | b^{t,d}, a^t) = \sum_{s \in S} b^{d,t-1}(s) p(s, s') \sum_{s' \in S} \prod_{i \in d} q_i(a_i^{t-1}, s', o_i^{t-1})$$

# 4. QUADRATIC PROGRAM

A policy in the sequence form is a solution to system of linear equations and inequalities. To be precise, a solution to the following system, based on (1)-(2), is a policy in the sequence form of agent $i$,

$$\sum_{a \in A_i} x_i(a_i) = 1 \quad (5)$$

$$-x_i(h) + \sum_{a \in A_i} x_i(h, o, a) = 0, \quad \forall h \in W_i \backslash Y_i, \forall o \in O \quad (6)$$

$$x_i(h) \geq 0, \quad \forall h \in W_i \quad (7)$$

This system consists of one variable $x_i(h)$ for each history $h \in W_i$. Thus, the variable $x_i(h, o, a)$ is for the history obtained on concatenating $o$ and $a$ to the history $h$. Let the size of $W_i$ be denoted by $n_i$. Let $m_i$ denote the number of equations in (5)-(6). Let $C_i$ denote an $m_i \times n_i$ matrix containing the coefficients of the left-hand sides of (5)-(6). Let $c_i$ denote an $m_i$-vector containing the right-hand sides of (5)-(6); thus, the first entry of $c_i$ is 1 and the remaining entries are 0s. Then, (5)-(7) can be written as $C_i x_i = c_i$, $x_i \geq \mathbf{0}$. Note that matrix $C_i$ is sparse (most of its entries are 0s).

The set of policies $X_i$ of agent $i$ is a polyhedron.

$$X_i = \{x_i \in \mathbb{R}^{n_i} | C_i x_i = c_i, x_i \geq \mathbf{0}\}$$

A joint policy an $n$-tuple of points, each point in a distinct polyhedron.

The discussion so far leads us directly to a linearly constrained quadratic program for the problem of multi-target tracking. In the multi-target tracking problem, in each configuration considered, there are only two agents in each subset $d$. Hence, the subsets in $D$ can be numbered as 12, 23, 34 etc. (12 means that agents 1 and 2 are in subset 12). Therefore, the expected reward (4) of a joint policy ($\sigma_1$, $\sigma_2$, …, $\sigma_n$) for this problem assumes a quadratic form. For example, for the 4-chain configuration, the expected reward is,

$$\sum_{J \in Y^{12}} r^{12}(J) \sigma_1(J_1) \sigma_2(J_2) + \sum_{J \in Y^{23}} r^{23}(J) \sigma_2(J_2) \sigma_3(J_3) + \sum_{J \in Y^{34}} r^{34}(J) \sigma_3(J_3) \sigma_4(J_4)$$

The expected reward of a joint policy can be expressed in matrix form as follows. Let the histories of each set $W_i$ be numbered from 1 to $n_i$. For the 3-chain configuration, $D = \{\{1, 2\}, \{2, 3\}\}$. Define an $n_1 \times n_2$ matrix $M^{12}$ whose rows are indexed by the histories of $W_1$ and whose columns by the histories of $W_2$, and whose $fg$th entry is,

$$M^{12}_{fg} = \begin{cases} r^{12}(f, g), & \text{if } f \text{ and } g \text{ are } \textit{both} \text{ terminal histories} \\ 0, & \text{otherwise} \end{cases}$$

Define an $n_2 \times n_3$ matrix $M^{23}$ analogous to $M^{12}$. Then, the expected reward of a joint policy ($\sigma_1$, $\sigma_2$, $\sigma_3$) for this configuration is,

$$\sigma_1' M^{12} \sigma_2 + \sigma_2' M^{23} \sigma_3$$

$\sigma'$ denotes the transpose of $\sigma$.

The expected reward of a joint policy for the other configurations can be similarly expressed in terms of matrices. For the 4-chain configuration, it can be expressed in terms

of matrices $M^{12}$, $M^{23}$ and $M^{34}$; for the 4-star configuration, in terms of matrices $M^{12}$, $M^{23}$ and $M^{24}$; for the 5-star configuration, in terms of matrices $M^{12}$, $M^{23}$, $M^{24}$ and $M^{25}$; for the 5-P configuration, in terms of matrices $M^{12}$, $M^{23}$, $M^{25}$, $M^{34}$ and $M^{45}$.

An optimal solution to the following quadratic program is an optimal joint policy for the 3-chain configuration,

$$\text{maximize} \qquad x_1' M^{12} x_2 + x_2' M^{23} x_3$$

subject to,

$$x_i \quad \in \quad X_i, \quad i = 1, 2, 3$$

A globally optimal solution $x_1^*$, $x_2^*$, $x_3^*$ to this QP is an optimal joint policy.

While this QP for the 3-chain configuration, its skeleton is applicable in fact to all the configurations of the problem. The only changes that are required to the program when moving from one configuration to another are to rewrite the objective function and to either add or remove sets of policy constraints (depending on whether the new configuration has more or less sensors than the previous configuration). For instance, an optimal solution to the following QP is an optimal joint policy for the 4-star configuration,

$$\text{maximize} \qquad x_1' M^{12} x_2 + x_2' M^{23} x_3 + x_2' M^{24} x_4$$

subject to,

$$x_i \quad \in \quad X_i, \quad i = 1, 2, 3, 4$$

The general form of the QP for the multi-target tracking problem is,

**(Q)**

$$\text{maximize} \qquad \sum_{d \in D} x_i' M^d x_{-i}$$

subject to,

$$x_i \quad \in \quad X_i, \quad i = 1, 2, \ldots, n$$

where for a given $d \in D$, $i$ and $-i$ represent respectively the indices of the two agents (sensors) that belong to $d \in D$.

PROPOSITION 1. *A globally optimal solution ($x_1^*$, $x_2^*$, …, $x_n^*$) to **Q** is an optimal joint policy.*

PROOF. By definition of a policy in the sequence form and the expected reward of a joint policy in the sequence form. □

As a algorithm, however, **Q** is not ideal because it is non-convex (in most cases). In other words, in most cases, none of the matrices,

$$\begin{pmatrix} 0 & -M^d \\ -M^{d'} & 0 \end{pmatrix}$$

is positive semi-definite. Solving **Q** is thereby guaranteed to yield only a locally optimal joint policy. In the next section, we convert **Q** to equivalent 0-1 mixed integer linear programs, solving which is guaranteed to yield an optimal joint policy.

# 5. MIXED INTEGER PROGRAMS

As stated in the opening, we present two different 0-1 mixed integer linear programs (MIPs) that are equivalent to **Q** in the sense that an optimal solution to the MIP is also a globally optimal solution to **Q**.

Both MIPs are based on the linearization of the objective of **Q**. Both MIPs yield an optimal joint policy that is *pure*, that is one in which each policy assigns conditional probabilities to histories that are either 0 or 1. The first MIP was described in [1] while the second MIP is novel.

The 0-1 MIP due to [1] is as follows.

**(M1)**

$$\text{maximize} \qquad \sum_{d \in D} m_d' z_d$$

subject to,

$$\begin{aligned}
& x_i \in X_i, \quad \forall i \in I \\
& l_{-i} x_i(h) = \sum_{J \in Y^d : J_i = h} z_d(J), \quad \forall i \in I, \forall d \in D_i, \forall h \in Y_i \\
& \sum_{J \in Y^d} z_d(J) = l_i l_{-i}, \quad \forall d \in D \\
& 0 \leq z_d(J) \leq 1, \quad \forall d \in D, \forall J \in Y^d \\
& x_i(h) \in \{0, 1\}, \quad \forall i \in I, \forall h \in Y_i
\end{aligned}$$

In this program, $m_d$ denotes a vector indexed by the terminal joint histories over $d$ (members of $Y^d$) and containing the expected rewards of these terminal joint histories, $l_i$ denotes $|O_i|^{T-1}$ and $D_i$ denotes the set of subsets in $D$ to which agent $i$ belongs to.

The program is a linearization of **Q** in that each quadratic term of the objective function of **Q** is replaced by a linear term (for instance, for $h \in Y_1$ and $\hat{h} \in Y_2$, $x_1(h)x_2(\hat{h})$ is replaced by $z_{12}(h, \hat{h})$). Thus, for each $d \in D$, $z_d$ is a vector of non-integer variables containing one variable per terminal joint history over $d$. For the 3-chain configuration, the variables of **M1** are thus the vectors $x_1$, $x_2$, $x_3$, $z_{12}$ and $z_{23}$.

The last line of the program ensures that the each $x_i$ is a pure policy. Placing 0-1 constraints on the variables representing terminal histories of each agent is sufficient to ensure that in every solution to the program, even the variables representing nonterminal histories of each agent acquire a value of either 0 or 1.

The constraints of **M1** are explained as follows. Assume we are given a pure joint policy ($\sigma_1$, $\sigma_2$, …, $\sigma_n$). Then: (1) The number of terminal histories of agent $i$ that receive a conditional probability of 1 from $\sigma_i$ is exactly $l_i$. (2) Therefore, the number of terminal joint histories over $d$ that receive a conditional probability of 1 from the joint policy ($\sigma_i$, $\sigma_{-i}$) is exactly $l_i l_{-i}$ (where $i$ and $-i$ are used to denote the two agents belonging to $d$) (3) Moreover, if a terminal history $h$ of agent $i \in d$ receives a conditional probability of 1 from $\sigma_i$, the number of terminal joint histories of which $h$ is a part of, and which receive a conditional probability of 1 from ($\sigma_i$, $\sigma_{-i}$) is exactly $l_{-i}$.

Note that in the program, we can replace the constraints,

$$l_{-i} x_i(h) \quad = \quad \sum_{J \in Y^d : J_i = h} z_d(J), \quad \forall i \in I, \forall d \in D_i, \forall h \in Y_i$$

by,

$$x_i(J_i) + x_{-i}(J_{-i}) - 2 z_d(J) \quad \geq \quad 0, \quad \forall d \in D, \forall J \in Y^d$$

without changing the set of outcomes of the program. However, the constraints of the latter type outnumber by far the constraints of the former type, and hence are not preferable.

PROPOSITION 2. *Given an optimal solution $(x_i^*)$, $\forall i \in I$, $(z_d^*)$, $\forall d \in D$, to **M1**, $(x_1^*, x_2^*, \ldots, x_n^*)$ is an optimal joint policy.*

PROOF. The proposition was proved in [1]; the proof is omitted here. □

We now move to the second 0-1 MIP. Recall that $D_i$ denotes the set of subsets in $D$ to which agent $i$ belongs to. For a terminal history $h \in Y_i$, for a $d \in D_i$ and for a pure policy $\sigma_{-i}$ define,

$$m_i^d(h, \sigma_{-i}) = \sum_{\hat{h} \in Y_{-i}} r^d(h, \hat{h}) \sigma_{-i}(\hat{h})$$

$-i$ denotes the other agent of the subset $d$. Furthermore, define,

$$m_i^{d-}(h) = l_{-i} \min_{\hat{h} \in Y_{-i}} r^d(h, \hat{h})$$
$$m_i^{d+}(h) = l_{-i} \max_{\hat{h} \in Y_{-i}} r^d(h, \hat{h})$$

$m_i^{d-}(h)$ and $m_i^{d+}(h)$ are respectively the lower and upper bounds on $m_i^d(h)$ for *any* $\sigma_{-i}$,

$$m_i^{d-}(h) \leq m_i^d(h, \sigma_{-i}) \leq m_i^{d+}(h)$$

Now consider the following 0-1 MIP.

**(M2)**

$$\text{maximize} \quad \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i} \left\{ m_i^{d+}(h) x_i(h) + w_i^d(h) \right\}$$

subject to,

$$x_i \in X_i, \quad \forall i \in I$$
$$w_i^d(h) \leq m_i^d(h, x_{-i}) - m_i^{d+}(h) x_i(h) - m_i^{d-}(h)(1 - x_i(h)),$$
$$\forall i \in I, \forall d \in D_i, \forall h \in Y_i$$
$$w_i^d(h) \leq 0, \quad \forall i \in I \; \forall d \in D_i, \forall h \in Y_i$$
$$x_i(h) \in \{0, 1\}, \quad \forall i \in I \; \forall h \in Y_i$$

This program contains one variable $x_i(h)$ for each history $h$ of each agent $i$. It also contains one variable $w_i^d(h)$ for each history $h$ of each agent $i$, for each of the subsets $d$. Notice the absence of variables for *joint* histories in this program. The size of the program is exponential in $T$ but linear in $n$.

PROPOSITION 3. *Given an optimal solution $(x_i^*, w_i^*)$, $\forall i \in I$, to **M2**, $(x_1^*, x_2^*, \ldots, x_n^*)$ is an optimal joint policy.*

PROOF. Note that for each $i = 1$ to $n$ for each $h \in Y_i$ and for each $d \in D_i$,

$$w_i^d(h) = 0, \quad \text{if } x_i(h) = 0$$
$$w_i^d(h) \leq m_i^d(h, x_{-i}) - m_i^{d+}(h), \quad \text{if } x_i(h) = 1$$

Therefore, we can write,

$$w_i^d(h) \leq \begin{cases} 0, & \text{if } x_i(h) = 0 \\ m_i^d(h, x_{-i}) - m_i^{d+}(h), & \text{if } x_i(h) = 1 \end{cases}$$

This being so, in every *optimal* solution to **M2**, neither of the following two cases arise: (1) $w_i^d(h) < 0$ and $x_i(d) = 0$, (2) $w_i^d(h) < m_i^d(h, x_{-i})$ - $m_i^{d+}(h)$ and $x_i(h) = 1$ (since we are maximizing, $w_i^d(h)$ will take the largest feasible value instead of the smallest).

Hence we have that,

$$w_i^d(h) = \begin{cases} 0, & \text{if } x_i(h) = 0 \\ m_i^{d+}(h) - m_i^d(h, x_{-i}), & \text{if } x_i(h) = 1 \end{cases}$$

In effect, from the objective function of **M2**, there obtains,

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i} \left\{ m_i^{d+}(h) x_i(h) + w_i^d(h) \right\}$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i : x_i(h) = 1} \left\{ m_i^{d+}(h) x_i(h) + m_i^d(h, x_{-i}) - m_i^{d+}(h) \right\}$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i : x_i(h) = 1} m_i^d(h, x_{-i})$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i : x_i(h) = 1} \sum_{\hat{h} \in Y_{-i}} r^d(h, \hat{h}) x_{-i}(\hat{h})$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} \sum_{h \in Y_i} \sum_{\hat{h} \in Y_{-i}} r^d(h, \hat{h}) x_i(h) x_{-i}(\hat{h})$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{d \in D_i} x_i' M^d x_{-i}$$
$$= \sum_{d \in D} x_i' M^d x_{-i}$$

which is the same objective function as in **Q**. Note that, as stated before, for a $d \in D$ and an $i \in d$, $-i$ denotes the other (than $i$) agent in $d$. In other words, in maximizing the objective function of **M2** subject to the constraints on the $x_i$ variables, for $i = 1$ to $n$, we are effectively maximizing the objective function **Q** subject to the same constraints on the $x_i$ variables. □

The size of **M1** or **M2** can be reduced by identifying *unrealizable* histories, and not including variables and constraints for such histories in the program. A history of an agent is unrealizable if, given the initial state $b^*$, the probability that every joint history, of which the history is a part, occurs is 0. Formally, a terminal history $h$ of agent $i$ is unrealizable if,

$$\text{prob.}(h(o), \hat{h}(o) | h(a), \hat{h}(a), b^*) = 0, \quad \forall d \in D_i, \forall \hat{h} \in Y_{-i}$$

Here, as before, $-i$ denotes the agent other than $i$ in set $d$. $h(o)$ denotes the sequence of observations of $h$ and $h(a)$ denotes the sequence of actions of $h$. A nonterminal history $h$ of length $t < T$ of an agent is unrealizable if every history of length $t + 1$ of the agent, whose first $2t$ - 1 elements coincide with $h$, is unrealizable. Note that the size of a program can be further reduced by iteratively identifying and excluding *dominated* histories. A dominated history is a history that is provably not required to find an optimal joint policy (an equally good or better history exists). However, this iterated elimination procedure involves solving a series of linear programs, and this in turn can be very time consuming.

## 6. COMPUTATIONAL EXPERIENCE

We tested the two MIPs, **M1** and **M2**, on the four sensor configurations shown in Figure 1 for horizon 3. The

| Sensor Config. | **M1** | **M2** | GOA [4] | SPIDER [6] | SPIDER-ABS [6] |
|---|---|---|---|---|---|
| 3-chain | 1.125 | 3.73 | $\approx 10^3$ | $\approx 10^1$ | $\approx 10^1$ |
| 4-chain | 1.148 | 14.22 | $\approx 10^4$ | $\approx 10^2$ | $\approx 10^2$ |
| 5-P | 49.3 | $> 4000$ | $\approx 10^4$ | $\approx 10^4$ | $\approx 10^4$ |
| 5-star | 22.25 | 3035 | $\approx 10^4$ | $\approx 10^3$ | $\approx 10^3$ |

**Table 1: Time taken in seconds by exact algorithms for solving for horizon 3.**

| | $M^*$ | Solver | Time Taken (s) | $M$ |
|---|---|---|---|---|
| 3-chain | 226 | SNOPT | 0 | 120 |
| | | LOQO | 0.012 | 163 |
| | | LANCELOT | 0.26 | 163 |
| 4-chain | 338 | SNOPT | 0.01 | 70 |
| | | LOQO | 0.14 | 248 |
| | | LANCELOT | 1.31 | 248 |

**Table 2: Performance of Q for horizon 3.**

programs were coded in Java and were solved through the branch-and-bound-revised simplex method using ILOG CPLEX. Table 1 shows the time taken by the programs to find an optimal joint policy.

The time taken is inclusive of every computation involved: the calculation of expected rewards of histories, calculation of upper and lower bounds (in **M2**), the identifying of unrealizable histories, the setting up and solving of the program. Also shown in the table is the time taken (approximate, since precise figures were not available) by three existing exact algorithms for this problem.

We also tested **Q** on the 3-chain and the 4-chain configurations for horizon 3. **Q** was coded in the AMPL language and solved using three freely available QP solvers from the NEOS website[1]: SNOPT, LOQO and LANCELOT. The results, given in Table 2, are somewhat discouraging in that while the solvers quickly find a solution, they seem unable to find an optimal joint policy. In the table $M^*$ denotes the expected reward of the optimal joint policy (as found by **M1** and **M2**) while $M$ denotes the expected reward of the locally joint policy found by **Q**.

The computational experience is limited to horizon 3. Longer horizons seem out of reach. On the smallest configuration, 3-chain, **M1** took 885 seconds to solve for horizon 4. For the other configurations, in solving for horizon 4, the two programs either cannot be formulated in memory for want of space, or when they can be, take too long to be solved.

# 7. DISCUSSION

Central to our mathematical programming approach is the use of the sequence form of a policy. In finding an optimal joint policy in the sequence form, we find for each agent a conditional probability distribution over his set of histories. The size of the set of histories of an agent is exponential in $T$, and it is reduced substantially (by upto fifty percent in the configurations considered), when unrealizable histories are removed from it. Thus, the use of the sequence form enables us to conceive mathematical programs of a reasonable size (exponential in $T$: the number of variables and constraints in **Q** as well as in **M2** is exponential in $T$ and linear in $n$ while it is exponential in $2T$ and linear in $n$ in **M1**).

As an exact algorithm, **M2** is much smaller that **M1**. However, this advantage in size is not matched by a commensurate advantage in time; indeed, the opposite is seen. **M2** takes much longer to be solved than **M1**. Why does **M2** fail where **M1** succeeds when both are subject to the same solver (ILOG CPLEX)? The crucial advantage **M1** holds over **M2** is that the matrix formed by the coefficients of its constraints is *sparse* and the *symmetric*. A working definition of a sparse matrix is that it is a matrix in which the zeros in each row far outnumber the nonzeros in each row. By symmetry, we mean that the zero and nonzero entries in the matrix are arranged in regular patterns. In **M1**, a typical row consists of a minus one, a very small block of ones and a very large block of zeros. The sparsity and symmetry of **M1**'s constraints' matrix allows the revised simplex method (used in ILOG CPLEX) to efficiently (rapidly and using little space) solve the relaxation LP of **M1** because it reduces the number of arithmetic operations conducted over the tableau and allows a faster inversion of the matrix. When this is not the case, as in **M2** whose constraints matrix is neither sparse nor symmetric, ILOG CPLEX falters given the large size of the program. This is one, possibly partial, explanation. A fuller understanding of the problem faced in solving **M2** may be arrived at by examining the revised simplex method in detail.

To summarize, we have applied a mathematical programming approach to the problem of multi-target tracking, and have obtained encouraging results when compared to existing approaches. For the configurations of sensors considered, the problem of finding an optimal joint policy reduces to a quadratic program (**Q**). We have shown two ways in which **Q** can be converted to an exact algorithm. Given the central place occupied by quadratic programming in the domain of nonlinear programming, it may be possible to conceive other ways (other MIPs). Another matter of further investigation could be the conception of approximate algorithms using **M1** or **M2**. As briefly stated before, the size of the MIP can be reduced by identifying all unrealizable or dominated histories. We can thereby use the criteria of $\epsilon$-unrealizability or $\epsilon$-dominance to further whittle down the size of the program in a controlled manner.

# 8. REFERENCES


[1] R. Aras. Mathematical Programming Methods For Decentralized POMDPs. *Ph.D. Dissertation, Université Henri Poincaré, Nancy*, 2008.

[2] D. Koller, N. Megiddo, and B. von Stengel. Fast Algorithms For Finding Randomized Strategies In Game Trees. *STOC*, 1994.

[3] J. Marecki, T. Gupta, P. Varakantham, M. Tambe, and M. Yokoo. Not All Agents Are Equal: Scaling Up Distributed POMDPs For Agent Networks. *AAMAS*, 2008.

[4] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked Distributed POMDPs: A Synthesis Of Distributed Constraint Optimization And POMDPs. *AAAI*, 2005.

[5] F. A. Oliehoek, M. T. Spaan, and S. Whiteson. Exploiting Locality Of Interaction In Factored DEC-POMDPs. *AAMAS*, 2008.

[6] P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. Letting Loose A SPIDER On A Network Of


[1] http://neos.mcs.anl.gov/neos/solvers/index.html

POMDPs: Generating Quality Guaranteed Policies.
*AAMAS*, 2007.

# Particle Filtering Approximation of Kriegspiel Play with Opponent Modeling

Josh Bryan, Piotr Gmytrasiewicz, Antonio Del Giudice
Department of Computer Science, University of Illinois at Chicago
Chicago, IL, 60607-7053, USA
E-mail: jbryan6@uic.edu, piotr@cs.uic.edu, a.delgiudice@gmail.com

## ABSTRACT

Kriesgpiel, or partially observable chess, is appealing to the AI community due to its similarity to real-world applications in which a decision maker is not a lone agent changing the environment. This paper applies the framework of interactive POMDPs to design a competent Kriegspiel player. The novel element proposed here is a way to handle complexity in I-POMDPs by using the notion of quantal response developed in behavioral game theory. This allows us to model only one (or a few) prototypical opponents while representing a whole ensemble of possible levels of expertise of the opponent's evaluation functions. The model is used to predict the opponent's likely moves. The moves of our own player can then be computed based on these predictions. Furthermore, we explore practical considerations when approximating I-POMDP solutions. Due to the immense number of possible states in a game of chess, we approximate the distribution over states using a sampling of possible states. We show the practical viability of our approach by testing a Kriegspiel playing agent against human players.

## 1. INTRODUCTION

Kriegspiel is a chess variant belonging to the family of invisible chess that encompasses *partially observable* variants of the popular game. Playing Kriegspiel is difficult, first, because the player needs to maintain a belief over all possible board configurations. Second, the player needs to select his move, given his belief about the board configuration and given the likely responses of the opponent. Predicting the likely responses is crucial and has a long tradition in minimax approaches to fully observable games. Minimax assumes that the players are rational and have opposing preferences, which is common knowledge.[1] Further, minimax is applicable only to fully observable games. In partially observable games one needs to model not only the opponent's preferences, but also the opponent's belief about the board configuration, and possibly his belief about the original player. Further, the opponent's level of expertise may also be in question in realistic settings.

Our approach is based on interactive partially observable Markov decision process [9] (I-POMDPs). Like POMDPs, I-POMDPs provide a framework for sequential planning. However, they generalize POMDPs to multi-agent settings by including the models of the other agent in the state space.[2] The models are used to form an informed prediction of the other agent's actions, which is then used during the move selection. Given the complications of maintaining the beliefs over the board configurations in Kriegspiel, the need to include the possible models of the other player further adds to the difficulty. We argue that without opponent modeling some important aspects of the game are necessarily neglected. In particular, without modeling the state of belief of the opponent the impact of moves which have the effect of supplying the opponent with information cannot be taken into account. As should be expected, I-POMDPs reduce to classical POMDPs if there are no other agents in the environment.

In previous work Parker et. al. [11] use sampling to represent beliefs over the state of the board, and avoid modeling the opponent explicitly by assuming that it will move randomly. Though we also use sampling to represent belief states, we do not model the opponent as random. Russell and Wolfe [13] consider whether guaranteed wins exist in some end-game configurations. Since the wins are to be guaranteed the opponent's state of belief does not matter and need not be considered. Parker et. al.'s work is particularly relevant to our approach because it can be viewed as an approximation. More precisely, the assumption that the opponent responds by executing a random move is an approximation of having a more informed prediction of the opponent's action obtained using a model of the opponent's preferences and beliefs about the state. Of course, one may model the opponent on a more detailed level by considering how it may model the original player, and so on. In I-POMDP framework [9] the nesting of models may be infinite, but finitely nested I-POMDPs are approximations which guarantee that the belief updates and solutions are computable. In our discussion below we illustrate how, for example, the assumption that the opponent will respond randomly approximates the solution obtained based on an explicit model of the opponent.

One of the obvious sources of complexity of solving I-POMDPs, apart from complexity inherent in POMDPs, is the need for considering all possible models of the opponent, and, if further levels of nesting are used, all possible models the opponent may have of the original agent, and so on. This paper proposes the use of the notion of quantal response [4,

[1] We should remark that common knowledge is a very strong, indeed unrealistic, assumption. See, for example, [10].

[2] For simplicity we assume the presence of a single other player throughout the rest of the paper.

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

12] to ameliorate the complexity. Quantal response allows one to use one model of the opponent to represent a whole ensemble of models with similar characteristics. In the case of Kriegspiel chess, instead of modeling all possible levels of expertise and beliefs of the opponent one can explicitly consider only one prototypical (hopefully the most likely) model. The predictions of this prototypical model are probabilistically "perturbed" to account for the possibility that the actual opponent's characteristics may not match those of the explicitly considered model exactly, but that large discrepancies are unlikely.

Another source of complexity in Kriegspiel specifically is the high dimensionality of the state space. There are approximately $10^{50}$ possible states of the chess board[2]. Though the state space is significantly smaller at the beginning of the game, it grows exponentially with each move. This high dimensionality makes maintaining an exact representation of the probability distribution over states practically impossible. Furthermore, since we are considering an interactive state space which includes the beliefs of the opponent, each possible state also has an associated distribution of states that represent the opponent's beliefs. This nesting of distributions can be as deep as we want. To tackle this problem, we maintain a sample set of interactive states to approximately represent the distribution over the entire state space.

## 2. KRIEGSPIEL

We briefly summarize the rules of Kriegspiel following [5, 11, 13, 14]. The game of Kriegspiel involves two players, $i$ (which will stand for White), and $j$ (Black), and one *referee*. The two players can see only their own pieces; the opponent's pieces are not visible. The referee can see the pieces of both players. The rules governing moves of the pieces are analogous to those in chess. Every time a player, say $i$, proposes a move to the referee, the following happens:

- If the move is illegal (i.e., it does not comply with the rules of chess given all the pieces on the board), the referee announces "Illegal", and the player $i$ may propose another move.
- If the move is legal, it is executed, and the referee announces:
  - "Capture in $X$", if a piece is captured on square $X$.
  - "Check by $Y$" if Black is in check. $Y$ can have the values of Rank (row), File (column), Short Diagonal, Long Diagonal and Knight.
  - If Black has no legal moves: "Checkmate" if Black is in Check and "Stalemate" otherwise.
  - "Black (White) to move" or "Silent" if none of the above happens.
  - "Draw." If the game is drawn due to insufficient material on the board for a checkmate to occur, or due to the 50 move rule being invoked.

## 3. INTERACTIVE POMDPS

The I-POMPD framework [9] generalizes the concept of single-agent POMDPs to multi-agent domains. An I-POMDP of agent $i$ is

$$\textit{I-POMDP}_i = \langle IS_i, A, T_i, \Omega_i, O_i, R_i \rangle \quad (1)$$

where:

- $IS_i$, the *interactive state*, is the cross product of the set of *physical* states $S$ (in our case possible board configurations) and the set of possible models of the opponent, $j$. We will consider only *intentional* models (or types) here. A type of agent $j$, $\theta_j$, consists of its belief state $b_j$ and frame $\widehat{\theta}_j$. As we explain further in the next section, the interactive states allow $i$ to keep track of $i$'s belief about the board configuration and about $j$'s beliefs.
- $A$ is the cross product of the actions agent $i$ and opponent $j$ can make.
- $T_i$, the *transition function*, defined as $T_i : S \times A \times S \rightarrow \{0, 1\}$. Thus, we assume that Kriegspiel is a deterministic domain and that agents' actions are assumed to only influence the physical state part of the interactive state space (this is called a model non-manipulability assumption in [9].)
- $\Omega_i$ is the set of possible observations of $i$, here assumed to be the set containing all possible *referee*'s responses.
- $O_i$ is an observation function $O_i : S \times A \times \Omega_i \rightarrow \{0, 1\}$. We assume that the referee's responses are deterministic according to the rules of Kriegspiel explained above.
- $R_i$ is the reward function $R_i : S_i \times A \rightarrow \Re$. . Both Kriegspiel and chess associate a reward only with terminal states of the game and the win, lose or draw of each agent. Since we cannot search the game tree so far into the future, we use a board evaluation function to represent the utility of a state (i.e., the measure that the board configuration will lead to the agent's win.)

It should be noted here that in POMDPs, utility of a belief state is computed as the sum of the reward from each state $s$ in a belief state $b$. In chess, the board evaluation function does not represent the reward for being in a given state, but rather the expected utility given all states reachable from $s$. That is, the board evaluation function acts as an approximation to the expected utility of being in any state if it were computed as the result from a fully observable Markov decision process. Thus, the use of a chess board evaluation function amounts to an MDP approximation of the real value of a belief state.

There is little research on board evaluation functions for Kriegspiel, so we have chosen to use the board evaluation function from a standard fully observable chess engine. In fact, for reasons noted above, board evaluation may be inappropriate, but instead a different kind of evaluation function could be defined over entire belief states. For lack of a better alternative, our implementation uses the gray-matter[1] library for move generation and board manipulation, and therefore, we have adopted the gray-matter board evaluation function as well. Looking at the effect of different board evaluation functions would be an interesting direction for future research.

## 4. PARTICLE FILTERING IN I-POMDPS

Particle filtering is a Monte Carlo method for approximating a posterior distribution over states $x_t$ at time $t$ given a sequence of observations $y_{1:t}$, a prior belief over states $p(x_0)$, a transition model $p(x_t|x_{t-1})$, and an observation

model $p(y_t|x_t)$. We will summarize the general algorithm here, however it is discussed in detail in [8]. First, a set of particles is initialized by sampling $N$ particles from the prior belief distribution. That is for each particle $x_0^i$ where $i = 1, \cdots, N$

$$x_0^i \sim p(x_0)$$

Then, at each time step $t$, three updates are performed on the particle set. First, the particle set is projected using the transition model. That is, each particle is resampled according to:

$$x_t^i \sim p(x_t|x_{t-1}^i)$$

Second, each particle is assigned a weight $w_t^i$ according to the observation function.

$$w_t^i = p(y_t|x_t^i)$$

Finally, a new particle set is resampled from this weighted particle set according to the weights of each particle. Thus, particles with low weights (and therefore low probabilities) die out and particles with high weights (and therefore high probabilities) are reproduced.

This method has been adapted for filtering and policy generation in I-POMDPs[7, 6]. In I-POMDPs, each particle is not simply a sample from the state space, but rather a sample from the entire interactive state space. Each particle therefore includes a physical state $s$ and a model of $j$. An intentional model consists of $j$'s belief and its frame, $\widehat{\theta}_j$, which contains other relevant properties of $j$ (for example, $j$'s reward function [9].) The representation of agent $j$'s belief in each particle is itself another particle set. The particles are therefore nested within each other to the depth of modeling that is desired. Transition models for particles are based on probability distributions over the opponent's actions as well as a model of the opponent's observations.

Policy evaluation is performed by considering every action available to the agent. For each action, the particle set is updated as though that action was performed. The particle set is then updated according to each possible observation at that time step. This process is repeated to project the particle set out to all reachable belief states at a given time horizon. Each belief state is evaluated by summing over the utility of the interactive states and dividing by the number of particles.

## 4.1 Sample Impoverishment

Sample impoverishment in particle filters is a problem that arises when the diversity in the particle set becomes small due to high importance samples being statistically sampled very often[3]. This is particularly a problem in models that have a low process noise. Because of the determinism of Kriegspiel play, if the particle set does not represent a diverse enough selection of states, it is possible that a given observation will be inconsistent with all particles in the set. In this case, $w_t^i = p(y_t|x_t^i) = 0$ for all particles, and no particle can be selected for the next iteration.

To deal with this particular case, we have adopted a strategy similar to that used by Parker et. al. [11] in which a sample set is generated that is consistent with the last observation. When no particles are consistent with the current observation, we generate a set of particles from a random distribution. The distribution is random given some known arrangement of the observable pieces at $t - w$ where $w$ is the size of a history window. Then, the random particle set is run forward to be consistent with the last $w$ observations. This gives us a particle set that, though not necessarily an accurate representation of the true distribution over interactive states, is at least consistent with the last several observations.

## 4.2 Quantal Response Approximation

The notion of quantal response we use has been coined in the fields of behavioral game theory and experimental economics [4, 12]. The idea is that decision makers can rarely be assumed to be perfectly rational, and to compute their utilities and probabilities exactly. Hence, one may need to replace the postulate that a decision maker is surely going to maximize his expected utility with a postulate that likelihoods of various actions increase as the actions' expected utilities increase. The shape of this dependence quantifies the likelihoods of the decision maker's mistakes, as well as the errors inherent in the model of the decision maker.

Formally, the quantal response is defined by:

$$P(\alpha_j) = \frac{e^{\lambda U(\alpha_j)}}{\sum_{\alpha_j} e^{\lambda U(\alpha_j)}} \quad (2)$$

where $P(\alpha_j)$ is the probability for opponent $j$ to execute action $\alpha_j$ and $U(\alpha_j)$ is the expected utility of an action computed by the model. The parameter $\lambda$ quantifies the degree to which our model of $j$ is correct. The high values of $\lambda$ indicate that $j$ is unlikely to perform an act that does not maximize its expected utility, as computed by the model.

As we mentioned the quantal response is an approximation to explicitly representing a large, or even infinite, number of possible models of the other agent. The computational advantage of having only one prototypical model is obvious. Also important is how this approximation naturally expresses the fact that the likelihood of the prototypical model being far off is small. We believe that such intuitive approximations to complexities of modeling agents are necessary to make them applicable to real-life problems.

# 5. IMPLEMENTATION AND EXPERIMENT

We have implemented a prototype Kriegspiel playing agent based on the I-POMDP particle filtering discussed here. The agent is written in a mix of C++ and Ruby. Performance critical portions are written in C++ so they can be optimized for speed, while the remainder of the application uses Ruby for its flexibility and rapid development features. The agent was designed to be played via a telnet client similar to the way Internet Chess Servers (ICS) can be accessed.

For our experiments, we built five agents with different parameters to test. These agents differ in the depth of opponent modeling, and in the number of time-steps that the agent looks ahead. Two of the agents modeled their opponent as being perfectly random. These agents were "0 depth" models in that they did not have any intentional models of their opponent. The other two agents had an opponent model of one layer deep (i.e. agent $i$ models agent $j$). Also, two of the agents computed the utility of moves by projecting the particle set 1 time-step, while two of the agents projected the particle sets 2 time-steps ahead. Thus we have four agents formed by different combinations of look-ahead and modeling depth. The fifth agent is a perfectly random

| | Random | 1 ahead, 0 deep | 1 ahead, 1 deep | 2 ahead, 0 deep | 2 ahead, 1 deep |
|---|---|---|---|---|---|
| Random | - | 0 / 3 | 0 / 4 | 0 / 5 | 0 / 4 |
| 1 ahead, 0 deep | 3 / 3 | - | 1 / 8 | 0 / 3 | 0 / 3 |
| 1 ahead, 1 deep | 4 / 4 | 0 / 8 | - | 8 / 23 | 2 / 6 |
| 2 ahead, 0 deep | 4 / 5 | 3 / 3 | 11 / 23 | - | 0 / 2 |
| 2 ahead, 1 deep | 4 / 4 | 3 / 3 | 1 / 6 | 1 / 2 | - |

**Table 1: Results of the 66 games between each of the five kinds of agents. The numbers indicate the number of wins by the agent identified by the row header when playing against the agent identified by the column header. e.g. The agent with 1 move look ahead and 0 depth opponent modeling won 3 out of 3 games against the random agent, but only 1 out of 8 games against the "1 ahead, 1 deep" agent.**

| | Random | 1 ahead, 0 deep | 1 ahead, 1 deep | 2 ahead, 0 deep | 2 ahead, 1 deep |
|---|---|---|---|---|---|
| Random | - | - | - | - | - |
| 1 ahead, 0 deep | 0 / 3 | - | - | - | - |
| 1 ahead, 1 deep | 0 / 4 | 7 / 8 | - | - | - |
| 2 ahead, 0 deep | 1 / 5 | 0 / 3 | 4 / 23 | - | - |
| 2 ahead, 1 deep | 0 / 4 | 0 / 3 | 3 / 6 | 1 / 2 | - |

**Table 2: Draw and stalemate results of the 66 games between each of the five kinds of agents. The numbers indicate the number of draws and stalemates out of total games between pairs of agents.**

agent that will be used as a baseline. All intentional agents are modeled using a set of 200 particles.

## 5.1 Computer vs. Computer

We ran a series of games where computer agents were paired at random. The number of wins and games for each pairing is reported in Table 1, and drawn or stalemated games are tallied in Table 2. It is worth noting that the unusually high number of games between the "1 ahead, 1 deep" and "2 ahead, 0 deep" agent is due to a day of testing where the other three agents had been disabled. Though, more data points would be useful, there are some interesting trends that can be observed. First, the random agent failed to win a single game. The best it did was a draw against the "2 ahead, 0 deep" player. This is not particularly surprising, but demonstrates that this approach is at least viable.

A second interesting point is that the "2 ahead, 1 deep" agent, which should be theoretically the one with the greatest planning abilities, only lost two games. This should be expected, but due to the amount of time required to run a game against this agent, we don't have a large enough sample set to say conclusively that this is the strongest agent.

| Wins | Human | 1 ahead 1 deep | 2 ahead 1 deep |
|---|---|---|---|
| Human | - | 5 / 12 | 6 / 7 |
| 1 ahead, 1 deep | 4 / 12 | - | - |
| 2 ahead, 1 deep | 0 / 7 | - | - |

**Table 3: Results of the 19 games between humans and 2 kinds of agents. The numbers indicate the number of wins by the agent identified by the row header when playing against the agent identified by the column header.**

| Wins | 1 ahead 1 deep | 2 ahead 1 deep |
|---|---|---|
| Human | 3 / 12 | 1 / 7 |

**Table 4: Draw and stalemate results of the 19 games between humans and the 2 agent types. The numbers indicate the number of draws and stalemates out of total games between pairs of agents.**

## 5.2 Computer vs. Human

To test the agent against human opponents, we invited students in an AI course to play the game online while we logged results. We had students play 19 games of Kriegspiel with two different agent designs. Though the human player generally performed better, the artificial agent was able to win 4 out of the 19 games. The win and drawn game tallies are represented in Tables 3 and 4.

What is interesting and somewhat surprising about these results is that the agent that looked farther ahead actually performed worse than the more myopic agent. Though we don't have enough data points to draw a definite conclusion, we put forward two possible hypotheses to explain these results. One is that the agent with further planning was a considerably slower agent (approximately 30-60 seconds per move) than the myopic agent. This additional time forced the humans playing to take more time pondering their own moves and therefore made better moves.

Another potential explanation is that the myopic agent tended to make bolder moves since it did not consider the counter move by the opponent. These bolder moves tend to result in a large number of captures. It is possible that the human players tend to be overly risk averse when playing against the near sited agent.

# 6. CONCLUSIONS AND FUTURE WORK

We have demonstrated that a reasonably performing Kriegspiel agent can be created using particle filter based approximations to I-POMDPs. We have also demonstrated a method by which a particle filter which has failed due to sample impoverishment can recover and continue to be useful, if not entirely precise. Furthermore, we have also shown that explicitly modelling the opponent's actions can perform better than assuming they are random.

The limited foresight of this implementation is a serious weakness, and future research needs to be spent tackling performance issues so that longer time horizons can be considered. Our future work will involve expanding this model to allow deeper lookahead. We will also explore other ways to resample particles when the sample set becomes impoverished.

Furthermore, more in depth testing should be done. We intend to open the Kriegspiel server up to a wider audience and invite experienced Kriegspiel players to play it. Also, comparison of play between agents with different parameters (number of particles, depth of nesting, length of time horizon, and type of board evaluation function) would be useful in determining what aspects make for a good Kriegspiel agent.

# 7. REFERENCES


[1] gray-matter - google code. http://code.google.com/p/gray-matter/.

[2] L. V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. Rijksuniversiteit Limburg; University Library, Maastricht University [Host], 1994.

[3] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[4] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

[5] P. Ciancarini, F. DellaLibera, and F. Maran. Decision making under uncertainty: a rational approach to kriegspiel. *Advances in Computer Chess 8*, 1997.

[6] P. Doshi and P. Gmytrasiewicz. Approximating state estimation in multiagent settings using particle filters. In *Proceeding of AAMAS 2005*, 2005.

[7] P. Doshi and P. Gmytrasiewicz. A method for approximating interactive pomdps using particle filters. In *Proceeding of AAAI 2005*, 2005.

[8] A. Doucet, N. D. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[9] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005. http://jair.org/contents/v24.html.

[10] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, July 1990.

[11] A. Parker, D. Nau, and V. Subrahmanian. Game-tree search with combinatorially large belief states. *IJCAI*, 2005.

[12] K. Ravikumar, A. Saroop, H. K. Narahari, and P. Dayama. Demand sensing in e-business, 2005.

[13] S. Russell and J. Wolfe. Efficient belief state and-or search, with application to kriegspiel. *IJCAI*, 2005.

[14] Wikipedia. Kriegspiel (chess) — wikipedia, the free encyclopedia, 2005. [Online; accessed 15-January-2006].

# Value of Communication In Decentralized POMDPs

Alan Carlin and Shlomo Zilberstein
Department of Computer Science
University of Massachusetts
acarlin@cs.umass.edu, shlomo@cs.umass.edu

## ABSTRACT

In decentralized settings with partial observability, agents can often benefit from communicating, but communication resources may be limited and costly. Current approaches tend to dismiss or underestimate this cost, resulting in overcommunication. This paper presents a general framework to compute the value of communicating from each agent's local perspective, by comparing the expected reward with and without communication. In order to obtain these expectations, each agent must reason about the state and belief states of the other agents, both before and after communication. We show how this can be done in the context of decentralized POMDPs and discuss ways to mitigate a common myopic assumption, where agents tend to overcommunicate because they overlook the possibility that communication can be deferred or initiated by the other agents. The paper presents a theoretical framework to precisely quantify the value of communication and an effective algorithm to manage communication. Experimental results show that our approach performs well compared to other techniques suggested in the literature.

## 1. INTRODUCTION

In multiagent settings, each agent is faced with three types of uncertainty. The first is uncertainty about the effects of its actions. This uncertainty is often addressed using the Markov Decision Process (MDP). The agent's world consists of *states*, and the agent's *actions* have probabilistic outcomes that change the state. The agent can receive a *reward* for entering a desirable state. The second type of uncertainty is about the state that the agent is in. This uncertainty can be addressed by adding *observations* to the model. The agent can reason about its state by combining its knowledge about state transitions with knowledge of its past actions and observations. The third type of uncertainty is about the state that the other agents are in, and the future actions that they will take, while accounting for the fact that the other agents perform similar reasoning. In this paper, we consider the Dec-POMDP (Decentralized Partially Observable MDP) model [2], and how this third type of uncertainty manifests itself within a Dec-POMDP.

One way to alleviate the latter type of uncertainty is to communicate with the other agents. In fact, it would usually be unrealistic to assume that agents do not communicate in a cooperative setting. But assuming ubiquitous communication is unrealistic for two reasons. First, trivially, if communication were ubiquitous, then in fact the problem could be solved and executed by one *centralized* agent, removing a key feature of multiagent systems. Second, in the real world communication is often not ubiquitous. Agents may be separated by distance, or the bandwidth between them may be limited, or they may operate in a low power environment where energy must be conserved. A common approach to factor this into the model is to assign communication a negative reward or *cost*.

In this paper, we will use the *sync* model of communication [13]. That is, when one agent decides to communicate, the result will be that all agents mutually exchange all available information. Because we assume that agents can synchronize in this manner, the paper studies the question of *when* to communicate. There is a rich, separate branch of the literature that studies *what* to communicate as well [9].

The paper proceeds as follows. First we discuss previous work on communication. Then we discuss the specific model that we use to produce communication decisions. An algorithm is developed that converts the complicated multiagent domain into a Hidden Markov Model in order to estimate the state of the other agents. The algorithm is expanded so that each agent can account for the communication policies of the other agents as well as their states. Finally, we show that the resulting planner performs well empirically.

## 2. RELATED WORK

The literature on communication can be divided into works that start with a centralized plan and those that do not. In the former group, agents generate a centralized policy at planning time, and then at execution time they communicate to enforce execution of the centralized plan. Xuan et al. consider the view of a "monitoring agent" whose knowledge consists only of jointly observable information since the last synchronization time [13]. Agents communicate whenever the monitoring agent notices ambiguity in what an agent should plan next. Roth et al. use the *tell* model of communication instead of the *sync* model [8]. Each agent uses its local history and the $Q_{\text{POMDP}}$ heuristic to reason about the joint action that should be taken. The history is also used to reason about communication.

Other works do not start with a centralized policy. Nair et al. introduce the Communicative DP-JESP (Dynamic Programing Joint Equlibrium-Based Search for Policies) technique, which integrates a communication strategy into $K$-step pieces of the JESP algorithm and finds a Nash equilibrium of policies for multiple agents [6]. In order to keep the algorithm tractable, the authors enforce a rule that communication must occur at least every $K$ steps.

Some recent work explores the concept of delayed communication. Spaan et al. find the best domain-level policy given that communication delays are stochastically possible [7].

The above approaches do not explicitly represent any cost to communicating. Overcommunicating is thought to be undesirable, either out of general principle, or because it can add to planning

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

time. Williamson et al. compute an explicit reward for communicating [12]. They introduce the *dec_POMDP_Valued_Com* model, which includes a communication reward function. Reward for communicating is based on the KL Divergence in the agents' belief states.

The approach most similar to ours has been developed by Becker et al. [1]. Communication incurs a negative reward, determined by the domain. Each agent determines the *Value of Communication* (VoC), which is the difference between the expected value of future policies with and without communication. However, the technique assumes that the world has joint full observability, that each agent fully observes its own local state, and furthermore that the other agents cannot affect its transitions or observations. The only interaction between the agents is via the joint reward function. The resulting problem is "only" NP-Complete [4], as the elimination of observations means that each agent only needs to reason about the global state, and not the belief states or observation histories of the other agents. In this paper, we will use a similar methodology to solve instances of the more complicated Dec-POMDP-Comm model, where each agent receives partial observations, and the agents are not transition or observation independent. Computing the value of communication in this more general context is substantially more complicated and is one of the key contributions of this paper. We retain the *sync* model of communication, though, as we will see in the next section.

## 3. DEC-POMDP

A Dec-POMDP is a Decentralized Partially Observable Markov Decision Process with Communication [3]. It is defined by the following components:

- A set of agents numbered $1..n$
- $S$, the set of domain states.
- $b_0 \in \Delta S$, the initial belief state distribution.
- $A = \times_i A_i$ is the set of joint actions, where $A_i$ is the set of actions available to agent $i$. At each time step, agents take one joint action $\mathbf{a} = \langle a_1, .., a_n \rangle$.
- $T$, the transition model: $T(s'|s, \mathbf{a})$ is the probability of transitioning to state $s'$ given the previous state was $s$ and joint action $\mathbf{a}$ was taken by the agents.
- $R$, the reward function: $R(s, \mathbf{a})$ is the immediate reward for taking joint action $\mathbf{a}$ in state $s$.
- $\Omega_1..\Omega_n$, the sets of observations possible for each agent. Each agent $i$ receives only its own observation $o_i \in \Omega_i$ at each timestep. The vector of received observations is $\mathbf{o} = \langle o_1, .., o_n \rangle$.
- $O$, the observation function. It specifies joint observation probability $O(\mathbf{o}|s', a_1..a_n)$, the joint probability that every agent $i$ sees corresponding observation $o_i$ after the agents took joint action $\mathbf{a}$ causing a state transition to $s'$.
- $H$, the horizon, or number of steps, in the problem.

We add communication to the model. Each agent has the option to initiate communication before taking an action. We restrict this paper to the *sync* communication model, so the communication language simply allows transmission of the agents' action/observation histories before each action. Communication is instantaneous, a message is received without delay as soon as it is sent. We also include $C$, a fixed cost on each step of communicating these synchronization messages. The fixed cost of $C$ is incurred if *any* number of agents choose to communicate. Otherwise, if no agent communicates, they incur no penalty. This problem is NEXP-hard. Indeed, when communication is prohibitively expensive, the model becomes a Dec-POMDP with no communication.

Since the problem has a finite horizon $H$, we can use a policy tree to represent a non-communicative policy of an agent. In the policy tree representation, nodes represent actions and branches represent observations. Each agent $i$ follows its own policy tree generated at the last synchronization step, referred to as $\pi_i^0$ with its first action corresponding to the root at time $t = 0$, and its last action corresponding to the leaves. $\pi_i^0$ contains a number of subpolicies, each corresponding to an observation sequence as the tree is traversed. We refer to an observation sequence as $\bar{o}$ and the resulting subpolicy as $\pi_i(\bar{o})$. Note that if we know an agent's initial policy and its sequence history of observations, we can derive its sequence of actions. Furthermore, the next sections will show that the local history of an agent can be combined with Bayesian reasoning on the Dec-POMDP model and the initial policies of the other agents to form a belief about the histories of other agents. To summarize, each node of an agent's policy tree maps to:

- A unique action/observation sequence $\bar{o_i}$
- A future local subpolicy rooted at the node $\pi_i(\bar{o_i})$
- A belief about the global $S$ as well as the action/observation histories of the other agents.

We will use these mappings throughout the paper. Unless stated otherwise we will also assume some housekeeping on the part of the algorithms that we describe, that knowledge of $\pi_i(\bar{o_i})$ implies knowledge of $\bar{o_i}$.

Let $b(s)$ be a belief state, and let $q$ be a variable representing a successor state. Let $a_i$ and $a_{-i}$ be the root actions of policies $\langle \pi_i, \pi_{-i} \rangle$. Standard theory on Dec-POMDPs says that the value of a joint policy tree, $\langle \pi_i, \pi_{-i} \rangle$ at a given belief state is recursively defined as the expected sum of the rewards of the subpolicy trees. That is:

$$V(\langle \pi_i, \pi_{-i} \rangle, b) = \sum_{s,q,o_i,o_{-i}} \Big[ b(s) T(q|s, a_i, a_{-i}) \\ O(o_i, o_{-i}|q, a_i, a_{-i}) V(\langle \pi_i(o_i), \pi_{-i}(o_{-i}) \rangle, q) \Big]$$

The above equation says that the value of the joint policy at $b_0$ can be decomposed into cases where the root actions result in a transition to state $q$, resulting in observations $o_i$ and $o_{-i}$. The base case of the recursion occurs at the last step of a finite horizon problem, where value simply corresponds to the reward $R(s, a_i, a_{-i})$ of the last actions taken.

## 4. SOLUTION METHOD

In our method, plans and communication strategies are deterimined offline and stored for use at runtime. The planner starts by precomputing optimal joint policies without communication (any non-communicative planner which generates policy trees can be used for this step). It also precomputes non-communicative joint policies for various reachable belief states of horizons $1...T$ (more details on this are in the next section), and stores these policies and their value in a cache. It uses these to construct a cache function for reachable belief distributions on the global state, and at runtime the cache will be accessed by each agent through a function call:

$$\text{CACHE}_i(b(S), h) \rightarrow \langle \pi_i^*(b(S)), \pi_{-i}^*(b(S)) \rangle$$

where $i$ is the identity of the local agent accessing the cache, $b(S)$ is the belief state it wants to query, $h$ is the depth of the policy and

$\pi^*$ represents that the policy is specific to that belief state. It also optionally stores a mapping of some or all observation sequences to communications decisions (if these are not stored, they could be recomputed by the agent at execution time).

$$\langle b(S), \bar{o}_i \rangle \rightarrow \{true, false\}$$

where $\bar{o}_i$ is a vector composed of the observations agent $i$ has made on prior steps. At execution time, each agent follows its policy and its communications policy. Upon communication, it retrieves the appropriate policy from the cache for the discovered belief state. We note trivially that if agents' policies are known to each other, then a joint observation sequence $\langle \bar{o}_i, \bar{o}_{-i} \rangle$ also determines a unique action history, and a unique $b(S)$ can be constructed by starting at the initial belief state and performing a forward computation as in a POMDP.

Before each action, each agent must decide whether to communicate. To do this, it uses the *Value of Communication*. Let $P(q, \bar{o}_{-i} | \bar{o}_i, \langle \pi_i, \pi_{-i} \rangle, b_0)$ represent the probability of reaching state $q$ while the other agents receive observations $\bar{o}_{-i}$ after $|\bar{o}_i|$ steps, given a starting belief state $b_0$ with policies $\langle \pi_i, \pi_{-i} \rangle$, and local observations $\bar{o}_i$. (The computation of this probability will be deferred to the next section). Let $\langle \pi_i, \pi_{-i} \rangle$ be the joint policy before communication and $\langle \pi_i^*(b_h), \pi_{-i}^*(b_h) \rangle$ be the joint policy that results from communication and discovery of joint belief state $b_h$.

DEFINITION 1. *The Value of Communication (VoC) is the difference between the expected value when communicating and the expected value for remaining silent.*

$$VoC(\bar{o}_i, \langle \pi_i, \pi_{-i} \rangle, b_0) = \sum_q \sum_{\bar{o}_{-i}} P_{q,\bar{o}_{-i}} (V^* - C - V)$$

*where*

$$\begin{aligned} P_{q,\bar{o}_{-i}} &= P(q, \bar{o}_{-i} | \bar{o}_i, \langle \pi_i, \pi_{-i} \rangle, b_0) \\ V^* &= V^*(\langle \pi_i^*(b_h), \pi_{-i}^*(b_h) \rangle, q, t) \\ V &= V(\langle \pi_i(\bar{o}_i), \pi_{-i}(\bar{o}_{-i}) \rangle, q, t) \end{aligned}$$

*$b_h$ is the belief distribution at time $h$ given $\langle o_i, o_{-i} \rangle$ and $b_0$.*

To understand the above definition, consider the perspective of agent $i$. It has synchronized with the other agents and determined that they synchronized in belief state $b_0$, it knows that the other agents have been following policies $\pi_{-i}$ since then, and that it has observed $\bar{o}_i$ since synchronization. In order to contemplate the value of remaining silent, it must consider the joint probability that the other agents' have observed $\bar{o}_{-i}$, and that the real current state is $q$. If this is the case, it knows that the agents will continue along subpolicies $\langle \pi_i(\bar{o}_i), \pi_{-i}(\bar{o}_{-i}) \rangle$, and the value of staying silent is simply the value of the joint subpolicy from state $q$. If the agents do communicate, they will combine observations to form a new joint belief state $b_h$, and they will follow a new joint policy for the belief state, $\langle \pi_i^*(b_h), \pi_j^*(b_h) \rangle$. The new joint belief state does not affect the fact that the true state is $q$, and so it computes the value of the new joint policy for $q$.

For example, consider the well-known multiagent Tiger problem [5], after an agent has observed Tiger-Left. In order to evaluate the value of communicating, the agent must consider each scenario that occurs after communication, one of which is the (small) chance that the other agent has also observed Tiger-Left, that they use the combined observations to open the door on the right, but that the true state was Tiger-Right, resulting in a large penalty.

## 4.1 Estimating the Joint History

We now explain how $P(q, \bar{o}_{-i} | \bar{o}_i, \langle \pi_i, \pi_{-i} \rangle, b_0)$ is computed. There are three sources of difficulty in this computation: (1) the local agent's history of actions has affected the transition matrix of the global state; (2) the other agents have adjusted their actions based on their observation history, not the true state; and (3) each local agent only holds its own observations, not necessarily the observations of the other agents.

**Algorithm 1**: Find SSTs for other agents at current step

**input** : Synchronized Belief State b, Nonlocal Policies $Q_{-i}$, Local Observation History $\bar{o}_i$, Local Action History $\bar{a}_i$, steps
**output** : An array of SSTs, each containing the true state, the remaining policies of the other agents, and a probability

```
begin
    D, E ← arrays of StateSubTrees, initialized to empty
    for i = 1 to |S| do
        D[i] ← ⟨i, Q_{-i}, b(i), false⟩
    for step = 1 to steps do
        E ← empty
        for i = 1 to |D| do
            ā_{-i} ← the root actions of D[i].tree
            a_i ← ā_i[step]
            o_i ← ō_i[step]
            for s' = 1 to |S| do
                for o_{-i} = 1 to |Ω_{-i}| do
                    SST ← new SST
                    α ← (D[i].p) T(s, a_i, a_{-i}, s')
                    O(s, a_i, a_{-i}, o_i, o_{-i}, s')
                    if nonmyopic then
                        Lookup SST.comm
                        if SST.comm == true then
                            prune SST
                    SST.s = s'
                    SST.p = α
                    SST.Q = D[i].Q.subTrees[o_{-i}]
                    Add SST to E
        Merge SSTs with equivalent subpolicies
        Prune SSTs with p < threshold from E
        Normalize each SST.p in E
        D ← E
    return D
end
```

DEFINITION 2. *Let a State SubTree (SST), be a tuple $\langle s, Q, p, comm \rangle$, where s is a state, Q is a finite-horizon policy, p is a probability, and comm is a boolean.*

Algorithm 1 shows how $P(\bar{o}_{-i}, q | \bar{o}_i, \langle \pi_i, \pi_{-i} \rangle, b_0)$ is estimated. The algorithm takes as input initial belief state $b_0$, the action and observation histories of the current agent $i$, and the known policies of the other agents at $b_0$. It outputs a set of SSTs at the current time step, each SST assigns a probability to one world state, composed of the actual state and the current policy of the other agents. SSTs are computed in a forward fashion. The set of SSTs is initialized to contain one element for each nonzero entry in $b_0$, with its $p$ being its probability in $b_0$, and its $Q$ being the initial policies of the other agents. At each time step, the current set of SSTs are used to generate a new set. Each SST in the new set represents a joint action taken by the other agents, a joint observation received, and a global state transition from an old SST, resulting in the new SST's state and subpolicy. The forward probability $\alpha$ is the probability of the old SST times the probability that the other agents made this transition, given the local agent's knowledge of its own action and observation on that step.

We also take the opportunity to merge SSTs with the same subpolicy. That is, if two observation histories of the other agent lead to the same subpolicy, there is no need to distinguish the two cases. Formally, if there are two SSTs,

$$\langle s, Q, p1, comm \rangle$$

and

$$\langle s, Q, p2, comm \rangle$$

, they can be merged into a single SST

$$\langle s, Q, p1 + p2, comm \rangle$$

This can be particularly useful in practice, if the non-communicative plans were built by an algorithm such as IMBDP [11], which builds plans where only a limited set of subpolicies are generated, and different observations lead to the same subpolicy.

There are other augmentations that can be made to Algorithm 1 which are not explored in this work. (1) The cache can be smaller and only contain likely decision points. At run-time, when a non-cached state is encountered, the agent can either initiate an online computation, or it can use the joint-policy from the least (Manhattan) distant cached belief-state. (2) SSTs can be generated by sampling from agent histories, rather than direct computation.

THEOREM 1. *The problem of estimating $M.p$ has an equivalent Hidden Markov Model (HMM) representation. Furthermore the algorithm is correct:*

*Suppose agent $i$ calls Algorithm 1 with threshold $0$ at time $t$ after observing $\bar{o}_i$, and the algorithm returns a set $Z$ of SSTs. Then $\forall M \in Z$, if $M = \langle s, Q, p, comm \rangle$, then $p$ is the probability that the global state is $s$ and the other agents' policies on this step are $Q$ at time $t$.*

PROOF. We can convert the problem of estimating $M.p$ into an HMM, and then solve using the forward-backward algorithm [10]. Each state of the HMM corresponds to a global state and an observation history of the other agents (we use the fact that each joint observation history maps to a specific joint subpolicy such as Q). State transition probabilities of the HMM correspond to state transition probabilities of the Dec-POMDP, given the local agent's action histories, times the probability of making the last observation. The transition probability is zero if the new observation history can not follow from the old. That is, a state with an observation history $w_1 w_2$ can not transition to a state with an observation history $w_2 w_2 w_3$, but it can transition to a state with observation history $w_1 w_2 w_3$.

Given this transition model, it is clear through induction (with the base case consisting merely of $S$ when Algorithm 1 is initialized) that the forward computation used to generate the leaves in the last step of Algorithm 1 are the same as the steps used to generate the corresponding states in the HMM. □

We note that approaches similar to Algorithm 1 appear throughout the literature. For instance, [6] states that "The key insight in the multiagent case is that if the policies of all other agents are fixed, then the free agent faces a complex but normal single-agent POMDP". However, we are unaware of a specific equivalency proof to an HMM, and we are hopeful that such an equivalency can be used in future work to leverage the rich HMM literature in Decentralized POMDPs.

PROPOSITION 1. *Assume the agents synchronize at belief state $b_0$ and form policies $\langle \pi_i, \pi_{-i} \rangle$, and the cost of communication is $C$. Assume a myopic perspective (the local agent may communicate only once, and the other agent can not communicate at all). The error introduced when agent $i$ makes a single communication decision after observation sequence $\bar{o}_i$ is at most:*

$$P(\bar{o}_i | \langle \pi_i, \pi_{-i} \rangle, b_0) \cdot C$$

PROOF. Assuming Theorem 1, if an agent computes its expectation of communication and decides not to communicate, it can never be wrong in the expected case. However, if it decides to communicate, it may be making an error. Since $V^* \geq V$, that is, the policy after communication is always at least as good as the one before, the error on this case bounded by $C$, and weighing by the probability of encountering the case in the first place, we have $P(\bar{o}_i | \langle \pi_i, \pi_{-i} \rangle, b_0) \cdot C$. Note that we are only considering the single communication decision; not possibilities that involve multiple future communication actions. This analysis is considerably more complex and will be handled in future work. □

The communication may not be necessary, as there are cases where (1) the other agent may initiate communication in all of the necessary states, or (2) communication can be deferred to a future step when more information is known.

Note that the number of SSTs can grow exponentially in each step, in the worst case. In practice, however, the number only grows with reachable belief states, and often on real-world problems only a small number of observations will be possible on each step. In order to keep the algorithm tractable, the algorithm can optionally prune SSTs with low probabilities at each step.

## 4.2 Non-myopic Reasoning

Algorithm 1 does not take into account the communication policy of the other agents, nor does it take into account the fact that communication need not be immediate, it may be deferred to future steps. In this section, we discuss how we improve the algorithm past this *myopic assumption*. The algorithm can be improved in three ways, first by using the fact that other agents did not communicate since the last *sync*, second by using the fact that other agents can communicate in the present, and finally by using the fact that communication can be deferred to the future.

## 4.3 Other agents in the past

We can use the knowledge that the other agents have not communicated since the last synchronized state. To do this, we use the *comm* field in the SST structure. At planning time, each agent computes VoC given its possible observation sequences and synchronized belief states. If VoC is positive, it sets *comm* to true. The *comm* value is stored for this history.

As Algorithm 1 is executed, each SST represents one possible observation history of this agent, and its children represent a continuation of that history. If the *comm* field is set to true for a corresponding observation history, this means that the agents would have communicated at this point. But any agent executing Algorithm 1 knows that didn't happen, since the algorithm initialized at the last communication point. Therefore it is known that the observation histories represented by such an SST never occurred, and the SST can be pruned.

(As an aside, the algorithm only generates accurate probabilities for SSTs in the current step. SST probabilities from previous steps are not necessarily reflect the probabilities of those histories. This is due to the nature of the forward-backward algorithm. In the last step, only a forward computation is necessary, such as that provided in the algorithm. For past steps, however, backwards information from subsequent steps would be necessasry. Let this backwards probability be $\beta$, and define it to be the probability that an SST's policy $Q$ was executed from its state $s$ given future observations. Finding $\beta$ can be computationally complex, as each leaf of $Q$ must be evaluated. Therefore in implementation we limit the analysis of $Q$ to the next $h$ steps, and only computing $\beta_h$. For instance define $\beta_1$ for an SST with state $s'$ and whose policy $Q$ corresponds to an

observation history $\bar{o}_{-i}$ to be:

$$\beta_1 \leftarrow \sum_{s''} \sum_{\bar{o}'' \in O''} T(s''|\mathbf{a}, s')O(\bar{o}''|\mathbf{a}, s'')$$

where **a** is a joint action composed of the known local action for that step as well as the root action for each other agents' policies in the SST, and $O''$ is the set of joint observations which must include the known observation from the local agent's history.)

## 4.4 Nash equlibrium in the present

Having modeled the communication strategy of the other agent on past steps, we turn to modeling the present step. To do this, we find a Nash equilibrium of communication strategies by constructing a matrix. For the two agent case, each row of the matrix corresponds to the SSTs for one agent, and each column corresponds to the SSTs for the other agent (for the multiagent case, each dimension represents another agent). Entries in the matrix correspond to the VoC given the history represented by the corresponding joint history, multiplied by the probability of that joint history. Each agent has the ability to communicate or not to communicate given a history. Communicating after a history corresponds to turning a row (or column, for the other agent) "on" or "off". The value of a joint communication strategy is the sum of the "on" values in the matrix. The myopic strategy discussed in above sections corresponds to turning each row or column on if its entries sum to a positive number. However, this illustrates the flaw of myopia, it does not maximize the value of the whole matrix, only its individual rows and columns. Since the row agent and column agent are not coordinating, they may double count entries. We improve on this by finding the Nash equilibrium. The approach is similar to the one described in [1], except (1) The rows and columns and probabilities correspond to observation histories, not states. (2) To reduce time of computation, agents can only alter $K$ rows, where $K$ is a parameter specified by the users. The remaining rows are toggled through myopic computation.

## 4.5 Value of deferring communication

The value of deferring communication to the future can be computed. For a given SST, the value of delay is the reward achieved by not communicating on the current step, added to the expected reward after communicating on the next step. The immediate reward is $pR(s, \mathbf{a})$ and it is added to:

$$p \sum_{s', \mathbf{o}'} T(s'|\mathbf{a}, s)O(\mathbf{o}'|\mathbf{a}, s')\mathbf{V}(\langle \pi_i^*(b_{h+1}), \pi_{-i}^*(b_{h+1}) \rangle, s')$$

where $p$ is the probability associated with the SST, **a** is the joint action specified by continuing the current policy of the local agent and the SST, $s$ is the state in the SST, $\mathbf{o}'$ the next joint observation, and $b_{h+1}$ is the belief state that would result at the next step. $\mathbf{V}$ is used to represent the fact that VoC must be retrieved for the local agent's observation in $\mathbf{o}'$, and if it is positive then $\mathbf{V} = V^*$ and $b_{h+1}$ is the belief state that results from communication while if $\mathbf{V}$ is negative, $\mathbf{V} = V$ and the joint policy merely continues. To compute the value of delaying communication, the computation above is summed for all SSTs returned by algorithm 1. If the sum is greater than or equal to the value of communicating on the current step, the agent does not communicate. A new value of delay will be computed after the next action is executed. Because of this, it is possible that the decision to postpone communication will cascade across several steps.

# 5. EXPERIMENTS

| horizon | Cost | No-Comm | Periodic | VoC-NM |
|---|---|---|---|---|
| 3 | 0 | 5.19 | 11.3 | 12.5 |
| 3 | 5 | 5.19 | 5.46 | 7.99 |
| 3 | 10 | 5.19 | 5.19 | 6.03 |
| 5 | 0 | 4.92 | 26.2 | 26.2 |
| 5 | 5 | 4.92 | 6.3 | 9.14 |
| 5 | 10 | 4.92 | 4.92 | 5.62 |
| 8 | 0 | 9.00 | 41.8 | 41.8 |
| 8 | 5 | 9.00 | 12.3 | 24.3 |
| 8 | 10 | 9.00 | 9.00 | 10.6 |
| 10 | 0 | 9.4 | 53.2 | 53.2 |
| 10 | 5 | 9.4 | 12.87 | 22.7 |
| 10 | 10 | 9.4 | 9.4 | 11.9 |

**Table 1: Comparison of various communications strategies for the Tiger problem.**

| horizon | Cost | No-Comm | Periodic | VoC-NM |
|---|---|---|---|---|
| 5 | 0 | 59.6 | 78.7 (4.0) | 78.7 (4.0) |
| 5 | 15 | 59.6 | 64.3 (1.0) | 64.9 (.89) |
| 5 | 30 | 59.6 | 60.3 (1.0) | 64.1 (.80) |

**Table 2: Comparison of various communications strategies for the BoxPushing-5 problem. Parentheses show the mean number of communications for each simulation.**

We considered our algorithm, labeled *VoC-NM* (Value of Communication - Non-Myopic), as compared to the algorithms of *No Communication*, *Full Communication* (communicating on every step), as well as *Periodic Communication* on various domains from the literature. For this latter strategy, we ran an algorithm which communicated every $K$ steps, and we used results from the best value of $K$ from 1 to the horizon of the problem. Thus, *Periodic* will provably outperform *No Communication* and *Full Communication*, so we do not separately list results for full communication. Our algorithm was implemented as follows: we precomputed values of communication for each agent for reachable histories at planning time by running a large number of simulations, and then stored this in a cache. We used a pruning threshold of 0, as we did not prune SSTs. We used the IMBDP planner [11] as the non-communicative submodule for this step. Then we ran a new 100,000 simulations of the non-myopic algorithm, referencing this cache on each simulation. Since MBDP-based planners only store a handful of subpolicies for each horizon step (using the same subpolicies for various branches of the larger policy tree), this choice of planners kept the size of the cache smaller.

The Multiagent Tiger problem [5] was simulated with horizon 10. Results show that the $VoC-NM$ planner was able to successfully communicate for both lower and higher costs of communication. Not shown in the figure, there was also a smooth decline in the number of communications attempted by the *VoC-NM* planner as cost of communication increased. There was an average of $3.5$ synchronizations for each simulated run when the communication cost was $C = 5$, at $C = 10$ there was an average of .5 communications, and at $C = 15$ communication was rare, the average was $.04$ per simulation. Running time was 9 seconds for the precomputation, and 2 seconds for the $100,000$ simulated runs after that. We also ran a myopic variant of the *VoC* planner, it did not include the algorithm enhancements of Section 4.2. The result across all tests was an approximately $10\%$ decrease in score at $C$ equal to 5 or 10.

We also ran the larger BoxPushing problem [11] for horizon 5, a

problem in which the value of the generated centralized and decentralized plans only differ by 20. Still, results similarly show that a *VoC-NM* methodology outperformed the other strategies because it communicates less, resulting in a gradual decrease in value as communication cost gets higher. The time taken for BoxPushing-5 was 4300 seconds at the planning stage, and then .38 seconds to run each simulation at execution time.

Across all experiments, a simple communication policy such as *Periodic* can be adequate when communication cost is low, or when communication points can easily be picked from the domain. As the cost of communication cost gets higher, and agents are motivated to avoid communication if possible, the richer *VoC-NM* approach is required. Even assuming, as we did, that the best period can be determined, a periodic communicator is forced to either choose to not communicate at all, or else to overcommunicate. This was shown as the *VoC-NM* approach reduced the amount of communication by 10% on BoxPushing when communication cost was 15, and by 20% when cost was 30.

## 6. CONCLUSION

We have presented a general approach for reasoning about costly communication within the Dec-POMDP framework. Computing the value of communication is challenging because each agent receives different partial observations and must reason about the possible synchronized state of the system after communication. We have shown that computing the value of communication can be used effectively to determine the utility of communicating versus staying silent. The approach allows each agent to make an exact estimate of the state of the other agents. We implemented and tested this capability using several standard benchmark problems. The results show that our approach uses communication effectively and outperforms a naive algorithm based on periodic communication. One area not explored in this paper is the tradeoff between building the cache of new policies at planning time, versus building it at runtime. Our implementation used for experiments generated the full cache at planning time, and always accounted for all possible observation histories. In future work, it would be interesting to trade accuracy for speed. This can be done by pruning more improbable belief state histories for the other agents as the algorithm progresses. Furthermore, it may be possible to drop the requirement that the exact value of the post-communication policies be used. Instead, perhaps a quicker heuristic could be used, such as the value of the centralized policy. These techniques create practical, yet disciplined ways to manage communication in decentralized multiagent systems.

## 7. REFERENCES


[1] Raphen Becker, Alan Carlin, Victor Lesser, and Shlomo Zilberstein. Analyzing Myopic Approaches for Multi-Agent Communication. In *Computational Intelligence*, volume 25, 2009.

[2] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. In *Mathematics of Operations Research*, volume 27, pages 819–840, 2000.

[3] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of The Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 137–144. ACM Press, 2003.

[4] Claudia V. Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[5] Ranjit Nair and Milind Tambe. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 705–711, 2003.

[6] Ranjit Nair, Milind Tambe, Maayan Roth, and Makoto Yokoo. Communication for improving policy computation in distributed pomdps. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM Press, 2004.

[7] Matthijs T.J. Spaan, Frans A. Oliehoek, and Nikos Vlassis. Multiagent Planning under Uncertainty with Stochastic Communication Delays. In *Proc. of The International Conference on Automated Planning and Scheduling*, pages 338-345, 2008.

[8] Maayan Roth, Reid Simmons, and Manuela Veloso. Decentralized communication strategies for coordinated multi-agent policies. In *Multi-Robot Systems: From Swarms to Intelligent Automata, volume IV*. Kluwer Avademic Publishers, 2005.

[9] Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? Execution-time decision in multi-agent POMDPs. In *Proceedings of the Eighth International Symposium on Distributed Autonomous Robotic Systems*, 2006.

[10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[11] Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 2007.

[12] Simon A. Williamson, Enrico H. Gerding, and Nicholas R. Jennings. A principled information valuation for communications during multi-agent coordination. In *AAMAS Workshop on Multiagent Sequential Decision Making (MSDM)*, 2008.

[13] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *In Proceedings of the Fifth International Conference on Autonomous Agents*, pages 616–623. ACM Press, 2001.

# On the Value of Commitment Flexibility in Dynamic Task Allocation via Second-Price Auctions

Tinglong Dai
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213
dai@cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
katia@cs.cmu.edu

Guoming Lai
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213
guomingl@cmu.edu

Robin Glinton
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
rglinton@andrew.cmu.edu

## ABSTRACT

Motivated by multi-agent systems applications, we study a task allocation problem in a competitive environment with multiple self-interested autonomous agents. Tasks dynamically arrive to a contractor that oversees the process of task allocation. Tasks are auctioned to contractees, who submit prices they require to accept tasks. The agent with the lowest bid wins but is rewarded with the second-lowest price. Each agent, based on his own state, will decide whether to participate in the auction or not, and will decide the bidding price if he chooses to participate. If a busy agent wins a new task, he has to decommit from his current task and pay a decommitment fee.

We formulate the problem and derive structural properties of equilibrium strategies. We also provide heuristics that are practical for multiagent system designers. Issues related to system design are discussed in the context of numerical simulations. The contribution is that (a) we provide formal analysis of contractees' optimal strategies in a given dynamic task allocation system with commitment flexibility; (b) we study the value of commitment flexibility in the presence of different system parameters.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

## General Terms

Management, Economics, Experimentation

## Keywords

Commitment Flexibility, Task Allocation, Multirobot Systems, Second-Price Auction

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

## 1. INTRODUCTION

Task allocation, especially in competitive environments, is one of the most important issues in the field of multiagent systems. In contrast to a cooperative environment in which agents work toward a common objective, a competitive environment is characterized by agents who seek to maximize their individual social utilities. In a dynamic task-allocation problem, agents not only compete directly with each other, but also compete indirectly with agents over time through the opportunity cost of potentially losing more lucrative tasks to others. A successful mechanism design, therefore, should induce the agents to contribute to the achievement of their common goals while maximizing their individual utilities.

Our work is motivated by the fast-developing multiagent resource/task allocation applications, including large-scale disaster relief (e.g., [10]), automated resource allocation of multiple distributed UAVs (e.g., [15]), and task allocation among multirobot coordination (e.g., [5]). These problems share the following characteristics: 1) significantly dynamic environments full of uncertainties, which makes it difficult to adopt centralized, static control schemes because tasks with different priorities/values/durations might arrive unpredictably. In a disaster relief system, for instance, agents face uncertain, and time stressed environments that require timely, flexible response. 2) The state of the agent network evolves over time such that it is possible that not the same set of agents are active in the agent society at each time. These features make full commitment contracts (i.e., a contract is binding once it is made), as assumed in most of the existing task allocation literature, less lucrative in terms of both system performance and agent utilities.

Prior research (e.g., [12], [2], [13]) has shown that it is indeed beneficial to the system to provide agents with a certain degree of commitment flexibility, i.e., an agent is allowed to walk away from its contract by paying a decommitment penalty. Such flexibility is especially favorable in cases with multiple job types, wherein agents' incentive to drop low-value and/or time-consuming tasks can enhance the system's capability of handling tasks. Nevertheless, aligning individual utilities and system-wide objectives cannot happen by magic. The agents need to be well coordinated by setting

appropriate decommitment penalties: too great decommitment penalties will force agents to become timid and continue on less lucrative jobs; too small decommitment penalties would allow agents to decommit from their jobs too frequently and sacrifice system-wide performance.

Our study is based on the contract net (CNET) protocol ([16]). A CNET consists of four parts: (a) Problem recognition, (b) task announcement, (c) bidding, and (d) awarding the contract. While the formats of (a), (b), and (d) are usually given in a multi-agent system, an efficient and effective auction mechanism is crucial in (c). A second-price-sealed-bid auction scheme is widely adopted in various systems due to its ability to induce bidders to bid their true valuations. In a second-price auction, the bidder who places the highest bid is granted the contract, but will pay the second-highest bid. We adopt an "inverse" second-price auction, i.e., the agent who asks for the lowest reward is the winner and is paid the second lowest reward in the auction.

Our goal is to design an auction-based mechanism that effectively takes advantage of commitment flexibility so as to enhance the system's task handling capability. The first step, however, is to examine individual agents' equilibrium strategies given various environment settings, e.g., decommitment penalty, arrival pattern of incoming tasks, and reward mechanism. The paper is organized as follows: In Section 2, we present relevant research. Then in Sections 3 and 4, we describe the problem, formulate each agent's problem and validate structural properties of equilibrium strategies. In Section 5, we apply heuristics to compare system performance under different environmental variables and mechanism settings. Issues related to system design are discussed in light of the numerical simulations. Finally, we conclude and discuss future research possibilities.

# 2. RELEVANT LITERATURE

Several research papers in the area of task allocation are relevant to our research. Sandholm and Lesser [12] study a leveled-commitment game between a contractor and a contractee, each of which is reluctant to walk away from the contract first. The study focuses on contract design and does not consider dynamic arrivals of incoming tasks. Abdallah and Lesser [1] build a model that integrates different aspects of mediator decision-making into a Semi-MDP model. The model is essentially a centralized model wherein the *mediator* rather than the agents has the option of decommitting from an unfinished task in pursuit of a more lucrative one. Sarne et al. [14] study the problem where multiple self-interested, autonomous agents compete for dynamically arriving tasks through a Vickrey-type auction mechanism. Each agent exits from the system after he gets a task assignment. They introduce the model and formulate equations by which the agents determine their equilibrium strategies. An efficient algorithm is proposed to calculate the equilibria. However, their model does not allow for the possibility of decommitment because it assumes one-shot task assignment for each agent. Brandt et al. [4] explore an automated task allocation mechanism combining auctioning protocols and contracts with commitment flexibility. They do not provide a mathematically rigorous model, but instead use simulation experiments to show the benefits of commitment flexibility. In addition, they do not take the equilibrium behavior of each agent into consideration. Finally, their model does not capture the various trade-offs the agents face in a dynamic task allocation environment and thus provides only limited insights.

Our research enriches the literature in that it brings together dynamic task allocation and commitment flexibility, with mathematical formulations, structural results of equilibrium behaviors, and computational analysis. This has not been done in the existing literature. This study helps system designers achieve a better understanding of the way commitment flexibility helps enhance a decentralized system's performance through auction mechanisms among self-interested, autonomous agents.

# 3. PROBLEM DESCRIPTION

## 3.1 Assumptions

- There is one contractor and a number of homogeneous contractees denoted by $n = 1, 2, \cdots, N$. $N$ can be a random number with known probabilistic distribution. The contractees are homogeneous in the sense that they have the same capabilities in performing the incoming tasks.
- Each contractee's activity is confidential to other agents. This implies $N \geq 3$.
- We assume an infinite planning horizon with intervals of equal length denoted by $t = 1, 2, 3, \ldots$.
- Each task has a constant maximum reward $M$.
- The duration of each task, denoted by $L$, follows a discrete probabilistic distribution with minimum length $L_{\min}$ and maximum length $L_{\max}$.
- There is an entry fee for participating in an auction, denoted by $C$.
- The probabilistic distribution of the duration of incoming tasks, the costs and rewards, are public to all the agents. However, each agent's task allocation information is confidential to other agents.
- A contractee who wins a task will receive his due reward in full immediately[1].
- The cost for each agent to perform a task is $c$ per period, and will be incurred period by period.
- A contractee's total cost of each incoming task does not exceed the contractor's budget, i.e.,

$$c \cdot L_{\max} \leq M.$$

## 3.2 Sequence of Events

Each round of auction follows the subsequent procedure:

- A task arrives at the beginning of each period.
- The contractor observes the incoming task and makes an announcement about the task duration to all the contractees.

[1]We make this assumption primarily for conciseness of the subsequent POMDP formulation. This assumption is also made in [14]. Assuming this would not necessarily encourage contractees to walk away, since the decommitment penalty might be larger than total reward of a task. We will extend this assumption in Section 4.3

- Each contractee decides whether to participate in the subsequent auction or not. A bidding contractee submits his own bid in a sealed envelope to the contractor.

  An agent who has an incomplete task on hand is allowed to participate in the auction. However, if he wins an auction, he has to decommit from his current task and pay a decommitment penalty, denoted by $d$.
- The contractor compares contractees's bids and announces the outcome of the auction. The winner is the contractee whose bid is lowest; he is awarded with a reward given by the second-lowest bid. Both the submissions of bids and announcements of results are conducted in a confidential manner such that each contractee is only aware of his own bids/results.

# 4. MODELING AND STRUCTURAL ANALYSIS

Our goal is to design a decentralized system that effectively makes use of agents' commitment flexibility to enhance the overall system's ability to complete as many tasks as possible. As the initial step, we analyze each contractee's behavior in the face of the environment we described in the preceding section.

## 4.1 Formulation

A contractee's objective is to decide whether to participate in the auction or not, and, if yes, how much to bid, so as to maximize his total discounted reward over the infinite planning horizon. We formulate contractee $i$'s problem as a POMDP (Partially observed Markov decision process, see [7]) characterized by a tuple $< \mathcal{S}, \mathcal{A}_i, \mathcal{T}, \mathcal{R}_i, \Omega_i, \mathcal{O}_i >$, the elements of which denote state space, action space, state-transition functions, the reward functions, observation space and observation functions, respectively. Here $\mathcal{S}$ and $\mathcal{T}$ are defined across all the contractees.

1) $\mathcal{S} = \mathbb{Z}_+^N \times \mathbb{Z}_+$ contains all the contractees' state (since one contractee's outcome is affected by other contractees' state) and the duration of the incoming task. Contractee $i$'s state at the beginning of period $t$ is defined as $x_{it}$ such that $x_{it} = 1, 2, \ldots$ denotes time-to-go until the completion of the currently active job and $x_{it} = 0$ means that the contractee is idle. The duration of the task arriving to the system at time $t$ is denoted by $y_t$.

2) $\mathcal{A}_i = \{0, 1\} \times \mathbb{R}$ involves a two-fold decision: whether to bid or not, and if yes, how much to offer.

3) $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ defines state-transition function. Define $h_{it} = 0$ or 1 as a contractee $i$'s bidding result at time $t$. We see that

$$\Pr[x_{i,t+1}|x_{it} \neq 0] = \begin{cases} 1 & \text{if } x_{i,t+1} = x_{it} - 1, h_{it} = 0 \\ 0 & \text{if } x_{i,t+1} \neq x_{it} - 1, h_{it} = 0 \\ 1 & \text{if } x_{i,t+1} = y_t - 1, h_{it} = 1 \\ 0 & \text{if } x_{i,t+1} \neq y_t - 1, h_{it} = 1 \end{cases}$$

and

$$\Pr[x_{i,t+1}|x_{it} = 0] = \begin{cases} 1 & \text{if } x_{i,t+1} = 0, h_{it} = 0 \\ 0 & \text{if } x_{i,t+1} \neq 0, h_{it} = 0 \\ 1 & \text{if } x_{i,t+1} = y_t - 1, h_{it} = 1 \\ 0 & \text{if } x_{i,t+1} \neq y_t - 1, h_{it} = 1 \end{cases}$$

4) $\mathcal{R}_i = \mathbb{R}$ specifies contractee $i$'s reward given an action. Our definition follows that if contractee $i$ places a bid, then his one-stage reward would be the second lowest bid minus decommitment penalty (if he has an unfinished task) if his wins, and 0 if he does not.

5) $\Omega_i = \{0, 1\}^\infty$ contains $h_{it}, t = 0, 1, \ldots,$ which is the only signal that contractee $i$ uses to reason about other contractees' state.

6) $\mathcal{O}_i : \mathcal{S} \times \mathcal{A} \to \Pi(\Omega_i)$ is the observation function $\omega(a, s')$ specifing the transition probabilities of making each possible observation $o \in \Omega_i$ given that contractee's bidding decision and state in the next period. Solving the above-defined POMDP is by no means an easy task due to the existence of a great number of agents and requirement of large state space. However, given that the initial system state is known, our POMDP can be converted to the following MDP formulation. We study the contractee's decision at the beginning of period $t$. $V_t^i(x_{it}, y_t)$ is defined as the maximum total discounted reward discounted by $\delta$ from time $t$ until infinity, given $x_{it}$ and $y_t$.

We see that the transition probability of $h_{it}$ is exactly contractee $i$'s probability of winning if he bids at time $t$. Instead of using $h_{it}$ as part of contractee $i$'s state, we include it to estimate the probabilities of winning and thus expected total rewards in the future.

A contractee's bidding strategy $s_{it} = (\Delta_{it}(x_{it}, y_t), b_{it}(x_{it}, y_t))$ for an auction consists of two parts: $\Delta_{it}(x_{it}, y_t) = 1$ indicates contractee $i$ will participate in the auction, and $\Delta_{it}(x_{it}, y_t) = 0$ otherwise. $b_{it}$ is contractee $i$'s bidding price if he chooses to participate in the auction.

Let $s_{-it}$ denote the symmetric bidding strategy of all the contractees other than contractee $i$ at period $t$. Given $s_{-it}$, and assuming the system state follows its stationary distribution, contractee $i$ would be able to determine the probabilistic distribution of the number of participants in the auction, and his probability of winning for a given bid. His estimated expected maximum total discounted reward function can be written as

$$V_t^i(x_{it}, y_t; s_{-it}) \tag{4.1}$$
$$= \begin{cases} -c + \max_{b_{it}} \Big\{ \delta\tilde{\mathbb{E}} V_{t+1}^i(x_{it} - 1, y_{t+1}; s_{-it}) \ , \\ \quad -C + \omega(b_{it}, s_{-it}) \Big[ \tilde{\mathbb{E}}[z_t|b_{it}, s_{-it}] - d + \\ \quad \delta\tilde{\mathbb{E}} V_{t+1}^i(y_t - 1, y_{t+1}; s_{-i,t+1}) \Big] \\ \quad + [1 - \omega(b_{it}, s_{-it})]\delta\tilde{\mathbb{E}} V_{t+1}^i(x_{it} - 1, y_{t+1}; s_{-i,t+1}) \Big\} , \\ \qquad \text{if } x_{it} = 1, 2, 3, \ldots \\ \max_{b_{it}} \Big\{ \delta\tilde{\mathbb{E}} V_{t+1}^i(0, y_{t+1}; s_{-it}) \ , \\ \quad -C + \omega(b_{it}, s_{-it}) \Big[ -c + \tilde{\mathbb{E}}[z_t|b_{it}, s_{-it}] \\ \quad + \delta\tilde{\mathbb{E}} V_{t+1}^i(y_t - 1, y_{t+1}; s_{-i,t+1}) \Big] \\ \qquad + [1 - \omega(b_{it}, s_{-it})]\delta\tilde{\mathbb{E}} V_{t+1}^i(0, y_{t+1}; s_{-i,t+1}) \Big\} , \\ \qquad \text{if } x_{it} = 0 \end{cases}$$

where $\omega(b_{it}, s_{-it})$ is the estimated probability of winning the auction given $s_{-it}$ and $b_{it}$, when $x_n \geq 1$ and $x_n = 0$, respectively. $\tilde{\mathbb{E}}[z_t|b_{it}, s_{-it}]$ denotes the estimated expected second-lowest bid in the $t$th round auction given $b_{it}$ and $s_{-it}$.

Since we are dealing with a discounted infinite-horizon problem, each contractee's decision is not time sensitive. Let $L$ be the random variable denoting the length of an incoming task, the above equation could be rewritten as

$$V^i(x_i, y; s_{-i}) \tag{4.2}$$
$$= \begin{cases} -c + \max_{b_i} \Big\{ \delta\tilde{\mathbb{E}} V^i(x_i - 1, L; s_{-i}) \ , \\ \quad -C + \omega(b_i, s_{-i}) \Big[ \tilde{\mathbb{E}}[z|b_i, s_{-i}] - d \\ \quad +\delta\tilde{\mathbb{E}} V^i(y-1, L; s_{-i}) \Big] \\ \quad +[1 - \omega(b_i, s_{-i})]\delta\tilde{\mathbb{E}} V^i(x_i - 1, L; s_{-i}) \Big\} , \\ \quad \text{if } x_i = 1, 2, 3, \ldots \\ \max_{b_i} \Big\{ \delta\tilde{\mathbb{E}} V^i(0, L; s_{-i}) \ , \\ \quad -C + \omega(b_i, s_{-i}) \Big[ -c + \tilde{\mathbb{E}}[z|b_i, s_{-i}] + \\ \quad \delta\tilde{\mathbb{E}} V^i(y-1, L; s_{-i}) \Big] \\ \quad + [1 - \omega(b_i, s_{-i})]\delta\tilde{\mathbb{E}} V^i(0, L; s_{-i}) \Big\} , \\ \quad \text{if } x_i = 0 \end{cases}$$

We have the following propositions that help determine the form of optimal equilibrium strategies:

PROPOSITION 1. *When $x \geq 1$, $V^i(x, y; s_{-i})$ is decreasing in $x$.*

PROOF. Intuitively, it is easy to see that $V(x, y; s_{-i})$ is decreasing in $x$ since the agent is always better off with a shorter task duration. We can show this by induction. To begin with, we look at the finite-horizon problem defined in (4.2). Assume that there are $N$ periods in total. Define the terminal equation as

$$V^i_{N+1}(x_{i,N+1}, y_{N+1}; s_{-i,N+1}) = -cx_{i,N+1}.$$

The above equation suggests that in the final periods, each contractee, if busy, will continue to work unless the task is finished.

Clearly $V^i_{N+1}(x_{i,N+1}, y_{N+1}; s_{-i,N+1})$ decreases in $x_{i,N+1}$. Now suppose that $V^i_k(x_{i,k}, y_k; s_{-i,k})$ decreases in $x_{i,k}$, we see

$$\begin{aligned} &V^i_{k-1}(x_{i,k-1}, y_{k-1}; s_{-i,k-1}) \\ &= -c + \max_{bi,k-1} \Big\{ \delta\tilde{\mathbb{E}} V^i_k(x^i_{k-1} - 1, y_k; s_{-i,k-1}) \ , \\ &- C + \omega(b_{i,k-1}, s_{-i,k-1}) \cdot [\tilde{\mathbb{E}}[z|b_{i,k-1}, s_{-i,k-1}] - d \\ &+ \delta\tilde{\mathbb{E}} V^i_k(y_{k-1} - 1, y_k; s_{-i,k})] \\ &+ [1 - \omega(b_{i,k-1}|s_{-i,k-1})]\delta\tilde{\mathbb{E}} V^i_k(x_{i,k-1} - 1, y_k; s_{-i,k}) \Big\} \end{aligned}$$

is decreasing $x_{i,k-1}$ because, for any given $y_{k-1}, s_{-i,k-1}, b_{i,k-1}$, both of the following two functions are increasing in $x_{i,k-1}$:

$$\begin{aligned} &f_1(x_{i,k-1}; y_{k-1}, s_{-i,k-1}, bi, k-1) \\ &= \delta\tilde{\mathbb{E}} V^i_k(x^i_{k-1} - 1, y_k; s_{-i,k-1}) \\ &f_2(x_{i,k-1}; y_{k-1}, s_{-i,k-1}, bi, k-1) \\ &= -C + \omega(b_{i,k-1}, s_{-i,k-1}) \cdot \{\tilde{\mathbb{E}}[z|b_{i,k-1}, s_{-i,k-1}] \\ &- d + \delta\tilde{\mathbb{E}} V^i_k(y_{k-1} - 1, y_k; s_{-i,k})\} \\ &+ [1 - \omega(b_{i,k-1}|s_{-i,k-1})]\delta\tilde{\mathbb{E}} V^i_k(x_{i,k-1} - 1, y_k; s_{-i,k}) \end{aligned}$$

We have shown that $V^i_k(x_{i,k}, y_k; s_{-i,k})$ decreases in $x_{i,k}, k = 1, \ldots, N+1$ in the above finite-horizon MDP. Now let $N \to \infty$, we see that $V^i(x, y; s_{-i})$ is decreasing in $x$. □

PROPOSITION 2. *$V^i(x, y; s_{-i})$ is decreasing in $y$.*

PROOF. An intuitive explanation to this proposition is that a larger task duration increases a contractee's cost, i.e., total costs throughout the duration of the task, and possible decommitment fees incurred in order to switch to a shorter task. Strict proof can be done by induction and is similar to that of Proposition 1. □

## 4.2 Structural Forms of Equilibrium Strategies

In this section, we establish the contractees' optimal bidding strategies (to bid or not to bid, and the optimal bidding price). We also apply our analytical framework to study the special case in which commitment flexibility is not allowed.

Before we derive our optimal bidding strategy, we make the assumption that a contractee with a shorter remaining time to finish the current task would like to place a higher bid. This makes sense because the contractee, when facing a shorter $x_i$, has less incentive to switch to other tasks. Similarly, we assume that a contractee would like to place a higher bid for a task with longer duration. This makes sense since tasks with longer duration have higher opportunity cost for the contractees.

To find out a symmetric bidding strategy, each contractee assumes that the policy function of other contractees is stationary, and makes its decisions conditioned on the stationary distributions formed when all the contractees choose symmetric strategies. This assumption is usually referred to as "fixed-strategy assumption" and made primarily for simplicity of modeling. Athey and Segal [3] give a review of dynamic mechanism design literature, and find out it is a common practice to assume that information is independent across periods and each agent has little access to information over time about other agent's type. Hu and Wellman [6] show that it is reasonable to make such an assumption when studying contractee's learning behaviors in that it is easy to model and implement, and, in some cases, such a model outperforms more sophisticated models.

THEOREM 1. ***(Optimal Bidding Strategy)***

*(i) A participant (denoted by i) with time-to-go $x$ in the auction for a task of duration $y$, given the equilibrium strategies of other contractees, will place a bid in the amount $b^*(x, y; s_{-i})$ such that*

$$b^*_i(x_i, y; s_{-i}) = \begin{cases} \delta\tilde{\mathbb{E}} \left[ V^i(x_i - 1, L) - V^i(y - 1, L) \right] + d \\ \quad + \frac{C}{\omega(b^*_i(x_i, y; s_{-i}), s_{-i})}, \text{if } x_i = 1, 2, \ldots \\ \delta\tilde{\mathbb{E}} \left[ V^i(0, L) - V^i(y - 1, L) \right] \\ \quad + \frac{C}{\omega(b^*_i(0, y; s_{-i}), s_{-i})} + c, \text{if } x_i = 0 \end{cases} \tag{4.3}$$

*(ii) If $x_i \geq 1$, contractee i's willingness to bid is increasing in $x_i$.*

*(iii) A contractee's willingness to bid is decreasing in $y$.*

*(iv) A busy contractee will choose to participate in the auction if and if only $y \leq \ell(x)$, where $\ell(x)$ is increasing in $x$; An idle contractee will choose to participate in the auction if and only if $y \leq \ell_0$, where $\ell_0$ is a constant.*

PROOF. (i) Our model can be viewed as a correlated private-value second-price auction in which each contractee's remaining duration is a private signal. Facing the same set of environmental variables (e.g., $y$, $C$,

$c$, $d$), the only factor that separates one contractee's decision from another contractee's is only their own remaining duration. Applying existing auction theoretic results (see, for example, [8] and [9]), contractee $i$'s optimal bidding price makes him indifferent as to whether to participate in the auction or not. In addition, contractee $i$ bids his own true valuation, i.e., we might replace $E[\text{second}|b_i, s_{-i}]$ with $b_i^*$.

When $x_i = 1, 2, 3 \ldots$ The optimal bidding price $b_i^*$ must satisfy:

$$\begin{aligned}&\delta\tilde{\mathbb{E}}V^i(x_i - 1, L; s_{-i})\\ &= -C + \omega(b_i^*, s_{-i})\left[b_i^* - d + \delta\tilde{\mathbb{E}}V^i(y-1, L; s_{-i})\right]\\ &+ [1 - \omega(b_i^*, s_{-i})]\delta\tilde{\mathbb{E}}V^i(x_i - 1, y; s_{-i}) \qquad (4.4)\end{aligned}$$

which gives

$$\begin{aligned}b_i^*(x_i, y; s_{-i}) = \delta\tilde{\mathbb{E}}\left[V^i(x_i - 1, L) - V^i(y-1, L)\right] + d\\ + \frac{C}{\omega(b_i^*(x_i, y; s_{-i}), s_{-i})}. \qquad (4.5)\end{aligned}$$

Similarly, we see that

$$\begin{aligned}b_i^*(0, y; s_{-i}) = \delta\tilde{\mathbb{E}}\left[V^i(0, L) - V^i(y-1, L)\right]\\ + \frac{C}{\omega(b_i^*(0, y; s_{-i}), s_{-i})} + c.\end{aligned}$$

(ii) We have previously assumed that $b_i^*(x_i, y; s_{-i})$ decreases in $x_i$. What we need to do now is to verify that this actually holds.

Examine the right-hand side of (4.5), given $s_{-i}$ when $x_i$ increases, it follows from Proposition 1 that

$$\tilde{\mathbb{E}}\left[V^i(x_i - 1, L) - V^i(y-1, L)\right]$$

decreases. Since $b_i^*(x_i, y; s_{-i})$ decreases in $x_i$, $\omega(b_i^*, s_{-i})$ will increase (a larger bid implies a lower chance of winning), which yields a decreasing $\frac{C}{\omega(b_i^*(x_i, y; s_{-i}), s_{-i})}$. We then see that both the right-hand and left-hand sides of (4.5) are decreasing in $x_i$.

We have assumed that the highest reward that a contractee could receive from accomplishing a task is fixed, which means a large $b_i^*(x_i, y; s_{-i})$ would put a contractee in a disadvantaged position. Hence contractee $i$'s willingness to bid is decreasing in $x_i$.

(iii) Similar to the proof of (ii).

(iv) Combining (ii) and (iii), we see that contractee $i$'s willingness to bid is increasing in $x_i$ but decreasing in $y_i$, which is exactly what we need to prove. Since all the contractees are homogeneous, all the contractees must follow the same strategies, which completes the proof.

(i)–(iv) give the stationary, symmetric optimal equilibrium strategies for the repeated second-price auction game. □

*A Special Case: No Commitment Flexibility*

We have provided analysis for the general case. Now we study the special case in which commitment flexibility is disabled. We will see that its MDP formulation as well as optimal bidding strategies are in simplified forms.

THEOREM 2. *If decommitment is forbidden, at the beginning of period $t$, an idle agent's strategy is to not to bid unless $y \leq \hat{\ell}$, where $\hat{\ell}$ is a constant.*

PROOF. When full commitment is assumed, each contractee's problem can be formulated as the following MDP (we inherit most of the notation in (4.1) except that we have to define a state variable for each contractee, since a contractee must be idle to be eligible for participating in the auction):

$$\begin{aligned}V^i(y; s_{-i}) = \max_{b_i}\Big\{&\delta\tilde{\mathbb{E}}V^i(L; s_{-i}), -C + \omega(b_i, s_{-i})\cdot\\ &\left\{\tilde{\mathbb{E}}[\mathrm{z}|s_i, b_{-i}] - yc + \delta^y\tilde{\mathbb{E}}V^i(L; s_{-i})\right\}\\ &+[1 - \omega(b_i, s_{-i})]\delta\tilde{\mathbb{E}}V^i(L; s_{-i})\Big\}\end{aligned}$$

where $\omega(b_i, s_{-i})$ denotes the contractee's probability of winning the auction given $b_i, s_{-i}$.

We can then use similar argument as in the proof of Part (i) of Theorem 1 to show that it is optimal for the contractee not to bid unless $y$ is below a certain threshold. □

Theorems 1 and 2 establish the basis for our simulation-based study in the following section. Theorem 2 implies that, when only full commitments are assumed, the equilibrium strategy for each idle agent is to bid when the duration of the incoming task is equal to or lower than a fixed threshold, and reject all the tasks with durations longer than the threshold. In the contrast, Theorem 1 says that in an environment where commitment flexibility is enabled, agents have different thresholds of task durations as to whether to participate in the auction or not, depending on each agent's state. Such commitment flexibility can make the system more "friendly" to tasks with longer durations and thus can accommodate more tasks in a given period.

## 4.3 Extension: Different Reward Mechanisms

The model we presented above does not consider comparison of different reward mechanisms. Consider, for instance, that a contractee is not given his full reward immediately after he wins an auction. Instead, he has to finish his task to get the corresponding reward. Another possible mechanism is that the contractee might receive partial rewards period by period according to his progress.

Our POMDP model can be revised to accommodate such differences by adding a new variable $r_i$ denoting the remaining reward from the currently active task (if any) to a contractee's state. Threshold policy in terms of $x_i$, $y$ and $r_i$ can be shown in similar fashion and is omitted here.

# 5. NUMERICAL STUDY

## 5.1 Experiment Design

We have previously established the structure of optimal equilibrium strategies. To have a concrete understanding of the benefits of commitment flexibility under different settings, we design a set of comparative experiments.

We resort to heuristics in our numeric study, recognizing that it is difficult to calculate the exact optimal solutions implied by our MDP model due to the following two difficulties:

- The number of bidders in each auction is uncertain. In a traditional task-allocation problem with an auction

protocol, only idle agents would bid for the incoming task. Our framework, in contrast, allows every contractee, busy or idle, to participate in the auction as long as it is profitable for him to do so. What makes our problem even more challenging is the existence of entry fee.

- Each contractee's valuation of the same new task depends on his current state. We do not have a straightforward expression of the probability distribution of agents' valuation of an incoming task, as opposed to well-studied auction problems in the Economics literature (e.g., [11]). Furthermore, the distribution of each agent's progress is closely related to the equilibrium strategies of the agent. This further separates our paper from [14], in which an agent's valuation of the incoming task solely depends on his own capabilities and the state of the world.

Without commitment flexibility, we assume that each contractee will bid at the beginning of period $t$ if and only if

$$y_t \leq \rho \cdot (L_{\max} - L_{\min}) + L_{\min}$$

where $0 < \rho \leq 1$ is a parameter that determines each agent's bidding strategy. A bidding agent's bid, depending on the duration of the incoming task, will be

$$b(y_t) = (\delta - \delta^{y_t})\mathbf{E}_{y_{t+1}} V(y_{t+1}) + y_t c + \frac{C}{\omega'},$$

$\delta \approx 1$ further simplifies the bid as

$$b(y_t) = y_t c + \frac{C}{\omega'},$$

where $\omega'$ is only partial known but can be estimated as follows: each contractee uses his own prior probability of winning to determine his bidding prices, and then use the bidding history over a period of time to update $\hat{\omega}'$ iteratively.

With commitment flexibility, we assume that contractee $i$ will use the following strategy: to bid if and only if

$$y_t \leq \rho_1 \cdot (L_{\max} - L_{\min}) + L_{\min}$$

when the agent is idle; to bid if and only if

$$y_t \leq \rho_2 \cdot x_{it}$$

when the agent is busy. An idle contractee will place a bid in the amount of

$$b(0, y_t) = y_t c + \frac{C}{\omega_0},$$

while a busy contractee will place a bid in the amount of

$$b(x_{it}, y_t) = (x_{it} - y_t)c + d + \frac{C}{\omega(x_{it})}.$$

Both the values of $\omega_0$ and $\omega(x_t)$ can be obtained through an iterative update approach (similar to the way we learn $\omega'$).

## 5.2 Simulation Experiments

Our model fits into the situation marked by dynamically incoming tasks and relatively limited number of agents. Our default experimental parameters reflect such a feature with $N = 7$, $C = 10$, $M = 100$, $c = 5$ and $d = 45$. We further assume that $L$, the duration of each incoming task, is uniformly distributed between $L_{\min} = 2$ and $L_{\max} = 20$. We observe 1000 periods and assume that the discount factor $\delta \approx 1$. In each set of experiment, we change one parameter and keep all the other parameters unchanged.

Define $THC$ (Task handling capability) as

$$THC = \frac{\text{Number of completed tasks}}{\text{The maximum possible number of completed tasks}}$$

$THC$ measures a system's ability to handle tasks.

### 5.2.1 Different Decommitment Penalties

We want to find out the behavior of the system over different values of decommitment penalties. The results are shown in Figures 1 and 3. From our experimental design, we see that $0 \leq \rho_1^* \leq 1$ measures an idle contractee's willingness to bid: the larger $\rho_1^*$ is, the contractee is more willing to bid for an incoming task. Similarly, $0 \leq \rho_2^* \leq 1$ measures a busy contractee's willingness to participate in the auction for the incoming task, which makes it possible for him to switch to a more attractive task.

We observe from Figure 1 that when the decommitment penalty is set too low, contractees have the incentive to bid for most of the incoming tasks and always to decommit from the current task, which leads to low productivity (see Figure 3). When the decommitment penalty is set too high, contractees are discouraged to decommit from long tasks, which make the system's ability of handling tasks relatively low.

The results also enable us to look at the benefits of commitment flexibility. When commitment flexibility is not allowed, each contractee would choose $\rho^* = 0.7$, resulting in $THC = 73.72\%$ and a total discounted reward of 972. In contrast, when commitment flexibility is enabled, contractees would choose $\rho_1^* = 0.95, \rho_2^* = 0.65$, which leads to $THC = 85.13\%$ and a total discounted reward of 1194, or a 15% improvement in $THC$.

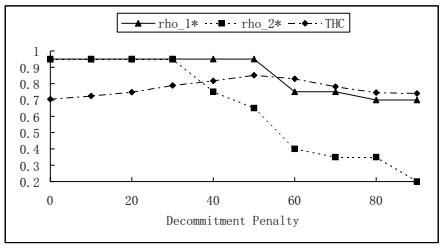


**Figure 1: $\rho_1^*, \rho_2^*$ and $THC$ for different decommitment penalties ($C = 10, c = 5, M = 100, L \sim [2, 20], N = 7$)**

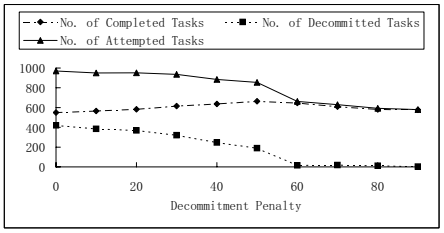


**Figure 2: Numbers of attempted, completed and decommitted tasks for different commitment penalties ($C = 10, c = 5, M = 100, L \sim [2, 20], N = 7$)**

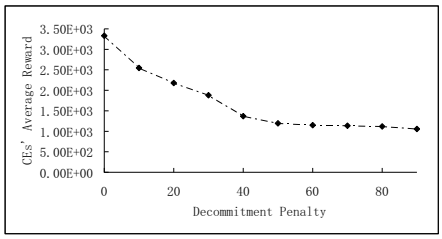


**Figure 3: Contractee's average reward for different decommitment penalties ($C = 10, c = 5, M = 100, L \sim [2, 20], N = 7$)**

### 5.2.2 Different Entry Fees

Choosing appropriate entry fee is also important in designing the auction mechanism. Our simulation results, as shown in Figures 4 and 5, reveal that a low entry fee will lead to low system productivity (as indicated by $THC$) because it encourages busy contractees to participate in auctions more frequently. When the entry fee is set to be too high, the system's task handling ability is inhibited due to contractees' limited participation.

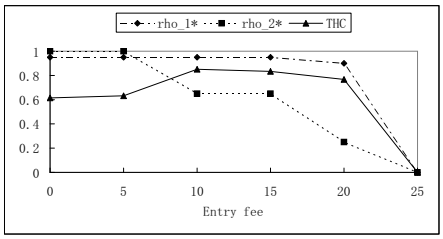


**Figure 4: $\rho_1^*, \rho_2^*$ and $THC$ for different entry fees ($d = 50, c = 5, M = 100, L \sim [2, 20], N = 7$)**

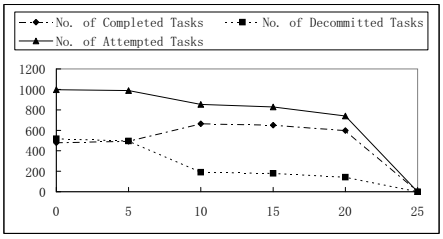


**Figure 5: Numbers of attempted, completed and decommitted tasks for different entry fees ($d = 50, c = 5, M = 100, L \sim [2, 20], N = 7$)**

### 5.2.3 Different Maximum Rewards Per Task

We observe from the experiment results (Figure 6) that a larger $M$ in general leads to high system productivity. If, however, the contractor's objective is not just to improve task handling ability, but also to reduce the total payments to contractees, the choice will be between cost and performance.

### 5.2.4 Different Task Duration Distributions

We fix $L_{\min} = 2$ and observe the changes brought by different $L_{\max}$'s. Our results (Figure 7) indicate that the system's ability of handling tasks decreases when $L_{\max}$ increases.

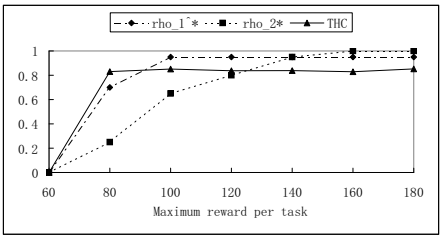


**Figure 6: $\rho_1^*, \rho_2^*$ and $THC$ for different $M$($d = 50, C = 10, c = 5, L \sim [2, 20], N = 7$)**

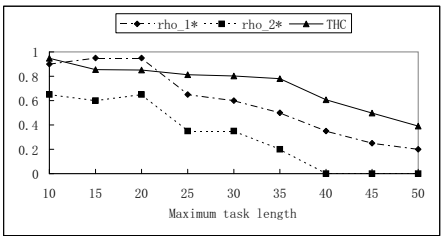


**Figure 7: $\rho_1^*, \rho_2^*$ and $THC$ for different $L_{\max}$($d = 50, C = 10, c = 5, M = 100, L \sim [2, L_{\max}], N = 7$)**

## 5.3 Discussions

Our experiments reveal the power of commitment flexibility. We might view $\rho$ or $\rho_1$ (as used in Section 5.1) as the threshold of an idle contractee when deciding whether to bid for the incoming task or not: a lower $\rho$ or $\rho_1$ means that the contractee is more "picky" about the duration of tasks, while a higher $\rho$ or $\rho_1$ makes it possible for relatively longer tasks to be put on the auction. When there is no commitment flexibility, the contractee would choose a relatively low $\rho$, after accounting for the opportunity cost brought by a long task. When a contractee is allowed to decommit from his current task, however, he would take a more relaxed manner when making an initial bid for a task, i.e., he can tolerate long tasks, because he still has a "second chance" made possible by commitment flexibility.

# 6. DISCUSSION AND FUTURE RESEARCH

In this paper, we study a dynamic task-allocation problem using repeated second-price auctions. We provide mathematical formulations and analysis, which lead to structural results of the equilibrium strategies. Our model is helpful for multi-agent system designers in revealing the benefits of commitment flexibility. In addition to the existing literature and our current work, more sophisticated models are needed. Below are several potential extensions of the current model that we plan to pursue:

*1. Different values.* Our model assumes a constant maximum reward for any incoming task. What would contractees' behavior vary if the incoming tasks possess different value? Such an extension can give us a more comprehensive understanding of the benefits of commitment flexibility.

*2. Bid on decommitted tasks.* Under our current problem settings, a decommitted task is discarded and cannot

be completed. Allowing contractees to bid on decommitted tasks would further improve the flexibility of the system. This is aligned with the notion of resale in the auction literature.

*3. Team work.* It would be interesting to consider the case when contractees can form teams to handle an incoming task. More interestingly, how does the team dynamics among contractees affect the contractor's decision of optimal bidding mechanism that induces contractees to complete as many tasks as possible? We leave this extension for future research.

## Acknowledgement

This research was funded by AFOSR grants FA95500810356, FA95500710039 and ARO grant W911NF0810301. We appreciate Alan Scheller-Wolf, Isa Hafalir and Praveen Paruchuri for their helpful comments in the preparation of this paper.

## 7. REFERENCES

[1] S. Abdallah and V. Lesser. Modeling task allocation using a decision theoretic model. Utrecht, Netherlands, 2005. AAMAS'05.

[2] M. Andersson and T. Sandholm. Leveled commitment contracts with myopic and strategic agents. *Journal of Economic Dynamics and Control*, 25:615–640, 2001.

[3] S. Athey and I. Segal. An efficient dynamic mechanism. Technical report, UCLA Department of Economics, 2007.

[4] F. Brandt, W. Brauer, and G. Weiss. Task assignment in multiagent systems based on vickrey-type auctioning and leveled commitment contracting. *Cooperative Information Agents IV, Lecture Notes in Artificial Intelligence*, 1860:95–106, 2000.

[5] M. Dias and A. Stentz. A comparative study between centralized, market-based, and behavioral multirobot coordination approaches. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, volume 3, pages 2279–2284, October 2003.

[6] J. Hu and M. P. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of the second international conference on Autonomous agents*, pages 239–246, Minneapolis, Minnesota, 1998.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[8] V. Krishna. *Auction Theory*. Academic Press, 2002.

[9] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[10] I. Nourbakhsh, M. Lewis, K. Sycara, M. Koes, M. Yong, and S. Burion. Human-robot teaming for search and rescue. *IEEE Pervasive Computing*, 4(1):72–78, Jan-Mar 2005.

[11] J. G. Riley. Expected revenue from open and sealed bid auctions. *Journal of Economic Perspectives*, 3:41–50, 1989.

[12] T. Sandholm and V. Lesser. Leveled commitment contracts and strategic breach. *Games and Economic Behavior*, 35:212–270, 2001.

[13] T. Sandholm and V. Lesser. Leveled-commitment contraction. *AI Magazine*, 23:89–100, 2002.

[14] D. Sarne, M. Hadad, and S. Kraus. Auction equilibrium strategies for task allocation in uncertain environments. *Lecture Notes in Computer Science*, 3191:271–285, 2004.

[15] P. Scerri, T. Von Goten, J. Fudge, S. Owens, and K. Sycara. Transitioning multiagent technology to uav applications. In *Proceedings of AAMAS 2008 Industry Track*, 2008. to appear.

[16] R. G. Smith. The contract net protocol high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

# Policy Iteration Algorithms for DEC-POMDPs with Discounted Rewards

Jilles S. Dibangoye
Laval University
Québec, Qc., Canada
gdibango@ift.ulaval.ca

Brahim Chaib-draa
Laval University
Québec, Qc., Canada
chaib@ift.ulaval.ca

Abdel-Illah Mouaddib
University of Caen
Caen, France
mouaddib@info.unicaen.fr

## ABSTRACT

Over the past seven years, researchers have been trying to find algorithms for the decentralized control of multiple agent under uncertainty. Unfortunately, most of the standard methods are unable to scale to real-world-size domains. In this paper, we come up with promising new theoretical insights to build scalable algorithms with provable error bounds. In the light of the new theoretical insights, this research revisits the policy iteration algorithm for the decentralized partially observable Markov decision process (DEC-POMDP). We derive and analyze the first point-based policy iteration algorithms with provable error bounds. Our experimental results show that we are able to successfully solve all tested DEC-POMDP benchmarks: outperforming standard algorithms, both in solution time and policy quality.

## 1. INTRODUCTION

In recent years, there has been increasing interest in finding scalable algorithms for solving multiple agent systems where agents cooperate to optimize a joint reward function, while having different individual observations. To formalize and solve such problems, [3] suggest a model that enables a set of $n$ agents to cooperate in order to control a partially observable Markov decision process. This framework can model environments under three constraints: *uncertainty*, *partial observability* and *decentralization*: uncertainty relies on the fact that the agents are imperfectly informed about action effects during the simulation; partial observability means that agents are imperfectly informed about the state of the process during the execution; and decentralization signifies that the agents are differently imperfectly informed during the execution. An environment that concurrently involves these three constraints is known as a decentralized partially observable Markov decision process (DEC-POMDP). Unfortunately, finding either optimal or even $\varepsilon$-approximate solutions of such problems has been shown to be particularly hard [12].

While some important progress has been made for solving finite horizon DEC-POMDPs, we still lack efficient algorithms with provable error bounds for the infinite horizon case. Indeed, the unique $\varepsilon$-optimal algorithm for the infinite horizon case runs quickly out of memory [2], as do optimal algorithms for the finite horizon case. This is mainly because they require the exhaustive enumeration of all possible joint policies at each time step, *i.e.,* the exhaustive backup. Unfortunately, the resulting set of joint policies requires an exponential space with respect to the number of joint observations and the number of agents. As a result, many attempts to solve infinite horizon DEC-POMDPs rely on memory-bounded algorithms [1, 4, 16]. These locally optimal algorithms use a fixed amount of memory, *i.e.,* the size of the solution is fixed prior to the execution of the algorithm. Even though the size of the solution is bounded, memory-bounded algorithms still suffer from the proved complexity of the problem. Moreover, choosing the right size for the solution is not obvious and dynamically adjusting it may raise non-negligible computation costs. Furthermore, though these algorithms tackle the space complexity efficiently, the time complexity remains too high and limits their ability to scale to medium solution sizes. Even more importantly, they fail to provide guarantees on the policy quality. Rather than constraining the size of the solution prior to the execution of the algorithm, it is equally possible to come up with a policy within a bound of the optimal policy.

In this paper, we design policy iteration (PI) algorithms that provide many desirable properties that current infinite horizon solvers lacked. First of all, we define exact and approximate Dynamic Programming (DP) backup operators that enable us to compute an improved value function. Secondly, we build up the joint policy based on the improved value function, this circumvents the problem of the exhaustive backup. Finally, we state and prove approximation error bounds on the resulted policy quality. The difficulty of this work lies in the definition of backup operators that guarantee: (1) the decentralization is preserved over the updates of the value function and the transformations of the corresponding policy while avoiding the exhaustive backup; (2) the updates of the value function are essentially DP updates. To leverage the first issue, we introduce new multi-agent concepts namely *basis objects* and *sets*, *i.e.,* partial joint information of the team that is sufficient to satisfy the decentralization. Even more importantly, these concepts help circumventing the problem of the exhaustive backup. To handle the second point, we perform essentially a single-agent DP update and keep track only on the value function that satisfies the decentralization.

## 2. BACKGROUND AND RELATED WORK

We review the DEC-POMDP model and the associated notation, and provide a short overview of the state-of-the-art algorithms.

### 2.1 The DEC-POMDP Model

DEFINITION 1. *A $n$-agent DEC-POMDP can be represented using a tuple $(I, S, \{A^i\}, P, \{\Omega^i\}, O, R)$, where: $I$ is a finite set of agents indexed by $1 \cdots n$; $S = \{s\}$ is a finite set of joint states; $A^i$ denotes a finite set of actions available for agent $i$, and $A = \otimes_{i \in I} A^i$ is the*

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

*set of joint actions, where $a = (a^1, \cdots, a^n)$ denotes a joint action; $P$: $S \times A \to \triangle S$ is a Markovian transition function. $P(s'|s,a)$ denotes the probability of transiting from state $s$ to state $s'$ when taking action $a$; $\Omega^i$ defines a finite set of observations available for agent $i$, and $\Omega = \otimes_{i \in I}\, \Omega^i$ is the set of joint-observations, where $o = (o^1, \cdots, o^n)$ is a joint observation; $O$: $A \times S \to \triangle\Omega$ is an observation function. $O(o|a,s')$ denotes the probability of observing joint observation $o$ given that joint action $a$ was taken and led to state $s'$; $R$: $A \times S \to \Re$ is a reward function. $R(a,s)$ denotes the reward signal received when executing action $a$ in state $s$;*

**Optimization criterion**. The DEC-POMDP model is parameterized by: $b_0 \in \triangle S$, the initial belief distribution, *i.e.,* the team belief over its initial state. The belief state (belief for short) $b \in \triangle S$ defines a probability distribution of the team over the underlying states. The next belief, denoted $b^{a,o} = \tau(b,a,o)$, that incorporates the latest joint action-observation pair $(a,o)$ and the current belief $b$, is updated as follows:

$$b^{a,o}(s') = \eta O(o|s',a) \sum_s b(s) P(s'|s,a) \quad (1)$$

where $\eta$ is a normalizing constant. When the agents operate over an unbounded number of time-steps, the DEC-POMDP has a discount factor, $\gamma \in [0,1)$. This model is coined by the term infinite-horizon DEC-POMDP with discounted rewards. Solving such a DEC-POMDP means finding a *joint policy* $\delta$ that yields the highest expected value $V^\delta(b_0) = \max_\delta \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau R(a_\tau, s_\tau)|b_0, \delta\right]$ where $V^\delta(b)$ denotes the expected sum of discounted rewards obtained given that joint policy $\delta$ is executed starting in belief $b$. Given the definition, the true value of a starting belief $b$ at time step $\tau$, which we write $V_\tau(b)$, is just $V^{\delta^*_\tau}(b)$ – where $\delta^*_\tau$ is an optimal policy at time step $\tau$. Since finding an $\varepsilon$-optimal joint policy is known to be intractable, we therefore state our optimization criterion as finding the best joint policy based on a small set of representative beliefs $B$ and the amount of time that is allotted to the algorithm. With this criterion, we want to design novel infinite horizon DEC-POMDP algorithms that update alternatively all, a belief set $B_\tau$, the optimal value function $V_\tau$ over $B_\tau$ and the corresponding joint policy $\delta_\tau$, until Bellman residual, *i.e.,* $\|V_\tau - V_{\tau-1}\|_\infty$, is less or equal to $2\varepsilon\gamma/(1-\gamma)$. That is the returned joint policy is an $\varepsilon$-optimal policy with respect to belief space $B$.

**Policy representation**. Throughout the paper, a policy for single agent $i$, *deterministic finite state controller* (DFSC), can be represented as policy graph $\delta^i = (X^i, \pi, \eta, x^i_0)$, where $X^i = \{x^i\}$ denotes a set of machine states; $\pi(x^i)$ is the individual action $a^i$ selected in machine state $x^i$; $\eta(x^i, o^i)$ is the successor machine state when individual observation $o^i$ is perceived in machine state $x^i$; and $x^i_0$ is the starting machine state. We denote a joint policy, *deterministic joint finite-state controllers* ( DJFSC ) by $\delta$. A DJFSC can also be represented as joint policy graph $\delta = (X, \pi, \eta, x_0)$, where: $X := \otimes_{i=1}^n X^i$ denotes a set of machine states; $\pi(x)$, $\eta(x,o)$ and $x_0$ are defined as for DFSCs. This contrasts with the standard representation based on policy vectors, *i.e.,* vectors of individual policies, one for each agent. Each joint machine state $x$ is associated to a hyperplane $\alpha_x$ – that is the vector value that denotes the expected sum of discounted rewards obtains when executing $\delta$ starting in machine state $x$.

## 2.2 Related Work

In this section, we first present POMDP relevant work that we use as a foundation for our PI algorithms. Then, we discuss the state-of-the-art $\varepsilon$-optimal approach to solving infinite horizon DEC-POMDPs.

**Policy Iteration for POMDPs.** A special case of DEC-POMDP, in which each agent shares its private information with its teammates at each time step, is called a multi-agent POMDP (MPOMDP). Because all the information available to the team at each time step is known, MPOMDPs can be solved using single-agent POMDP techniques. Over the past few years, exact and approximate PI algorithms have been proposed for POMDPs by [15, 6] and [8], respectively. These algorithms, summarized in Algorithm 1, share the same structure that consists of a threefold method: first, compute the value function $V_\tau$ of the current DJFSC $\delta_\tau$ (*policy evaluation*); secondly, update the value function $V_\tau$ represented by a set $\Gamma_{\tau+1}$ of hyperplanes into an improved value function $V_{\tau+1}$ represented by a set $\Gamma_\tau$ of hyperplanes (*policy improvement*); and finally, transform the current DJFSC $\delta_{\tau-1}$ into an improved DJFSC $\delta_\tau$ (*policy transformation*) with respect to $\Gamma_{\tau+1}$. These processes alternate until a convergence criterion is reached. The policy evaluation step is straightforward when a policy is represented as a DJFSC [6]. This can be achieved by solving the system of linear equations:

$$\alpha_x(s) = R(s,\pi(x)) + \gamma \sum_{s',o} P(s'|s,\pi(x)) O(o|s',\pi(x)) \alpha_{\eta(x,o)} \quad (2)$$

where $x$ and $\eta(x,o)$ are machine states. In addition, the policy improvement step relies on the fact that a POMDP can be reformulated as a belief MDP. Indeed, it is well-known that in POMDPs the belief is a sufficient statistic for a given history. Therefore, the value function $V_\tau$ can be updated using DP. [13] show how to implement the DP update of a value function $V_\tau$ by exploiting its piece-wise linearity and convexity. Because the value function is a mapping over a continuous $|S|$-dimensional space, $V_{\tau+1}$ cannot be directly computed. Instead, the corresponding set $\Gamma_{\tau+1}$ can be generated through a sequence of operations over $\Gamma_\tau$.

**Algorithm 1** Policy Iteration

```
1: procedure PI(initial policy: δ_0, belief space: B)
2:     Initialization: δ_{τ+1} = δ_τ = δ_0
3:     repeat
4:         % policy evaluation
5:         Compute Γ_τ ← {α_i}, i.e., value function of δ_τ
6:         % policy improvement
7:         Γ_{τ+1} ← BACKUP(Γ_τ, B) where B may be △S
8:         % policy transformation: δ_τ → δ_{τ+1}
9:         Re-initialization: τ = τ + 1
10:    until ‖V_{τ+1} − V_τ‖_∞ ≤ 2εγ/(1−γ)
```

**DP update.** We now describe the straightforward implementation of DP update for POMDPs [14]. First, we generate intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$ $\forall a, o$ (*Step 1*): $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = R(s,a)$ and $\Gamma^{a,o} \leftarrow \alpha^{a,o}_i(s) = \sum_{s'} P(s'|s,a) O(o|s',a) \alpha_i(s')$, $\forall \alpha_i \in \Gamma_\tau$. Next, we create $\Gamma^a$ ($\forall a \in A$), the cross-sum over joint observations, which includes one $\alpha^{a,o}$ from each $\Gamma^{a,o}$ (*Step 2*): $\Gamma^a \leftarrow \Gamma^{a,*} \oplus_{o \in \Omega} \Gamma^{a,o}$. Finally, we take the union of $\Gamma^a$ sets (*Step 3*): $\Gamma_{\tau+1} = \cup_{a \in A} \Gamma^a$. It is often the case that a hyperplane in $\Gamma_{\tau+1}$ is completely dominated by another hyperplane ($\alpha \cdot b \leq \alpha' \cdot b$, $\forall b$) or by combination of other hyperplanes. Those hyperplanes can be pruned away at each steps of the update without affecting the policy quality. Finally, [6] suggests a number of rules that enables us to transform the current policy $\delta_\tau$ into an improved one $\delta_{\tau+1}$. To the best of our knowledge, there is no known techniques that extend completely this approach to multiple agent settings.

**Policy Iteration for DEC-POMDPs.** [2] proposed recently an attempt to extend PI from single-agent into decentralized multi-agent settings. This algorithm builds over a series of steps a vector of stochastic finite state controllers, one for each agent. Each

step consists of a twofold method: the exhaustive backup and the pruning of dominated stochastic machine states. For each agent $i$, the exhaustive backup takes as input the current set of stochastic machine states $X^i_\tau$. Then, it builds new stochastic machine states $X^i_{\tau+1}$ for all possible joint actions, such that the successor links of the new stochastic machine state are associated only to stochastic machine states that appear in the current sets. The resulted set of stochastic machine states $X^i_{\tau+1}$ is exponential with respect to the number of observation, *i.e.,* $|X^i_{\tau+1}| = |A^i||X^i_\tau|^{|\Omega^i|} + |X^i_\tau|$. Thus, the joint stochastic controller grows by $\prod_{i=1}^n |A^i||X^i_\tau|^{|\Omega^i|}$, which grows exponentially with $n$. Thereafter, a pruning step eliminates dominated stochastic machine states without loosing the ability to eventually converging to an $\varepsilon$-optimal controller. Performing this pruning step, however, can be extensive since they require a linear program and its dual. While interesting, this PI algorithm laks many desirable properties of the PI algorithm in single-agent settings. Among many, the algorithm fails to define a DP backup operator for the decentralized multi-agent case. This would enable us to derive the Bellman residual $\|V_\tau - V_{\tau-1}\|_\infty$ – distance between two successive value functions and even more importantly approximation error bounds. Furthermore, as already discussed in the POMDP section, dominated stochastic machine states can be pruned away at earlier steps of the backup. Doing so, would avoid the exhaustive generation of all possible stochastic machine states which is expensive.

# 3. POLICY ITERATION REQUIEREMENTS

In order to extend PI algorithms along with its properties from single-agent to the decentralized multi-agent settings, we need to face two key issues. As we aim at finding a policy for DEC-POMDPs, we need to make sure that our backup operator $\mathbb{H}_B$ transforms set $\Gamma_\tau$ into an improved set $\Gamma_{\tau+1}$ while preserving the decentralization – that is all hyperplanes $\alpha \in \Gamma_{\tau+1}$ and the improved policy $\delta_{\tau+1}$ satisfy the decentralization constraint. On the other hand, to inherit POMDP properties our backup operator need essentially to be the DP backup operator – that is the value $V_{\tau+1}(b)$ at belief $b$ depends on values $V_\tau(b')$ of its successor beliefs $b'$: $V_{\tau+1}(b) = \mathbb{H}_B V_\tau(b) = \max_a \mathbb{E}\left[R(a,b) + \gamma \sum_o V_\tau(\tau(b,a,o))\right]$.

## 3.1 Satisfying the Decentralization

In this section, we introduce new multi-agent planning concepts namely *basis* objects and sets. These concepts help preserving the decentralization while updating hyperplanes or transforming policies.

**Preliminary definitions.** A set of basis joint observations also called *basis set* and denoted $\mathring{\Omega} \subseteq \Omega$ is a set of joint observations of the smallest size where each individual observation of any agent, *e.g.,* $o^i \in \Omega^i$, is included in at least one joint observation of the basis set, *e.g.,* $(\cdots, o^i, \cdots) \in \mathring{\Omega}$. Because all individual observations of all agents are represented in at least one joint observation of the basis set, the *cardinality of a basis set* $\mathring{\Omega}$, denoted $\kappa(\Omega)$ ($\kappa$ for short), is given by:$\kappa = |\mathring{\Omega}| = \max_{i \in I} |\Omega^i|$. This holds for any basis set and even more importantly for any number of agents. In the remainder of the paper, we assume that each agent has the same number of individual observations [1]. Therefore, a straightforward way of building a basis set $\mathring{\Omega}$ is to include the largest set of joint observations such that any pair of joint observations is *component-wise different*, *e.g.,* $(o_1, o_2)$ and $(o_2, o_1)$. A *basis object* is any object, *e.g.,* policy or hyperplane, defined only over basis observations. A *basis joint policy* $\delta_B \colon \mathring{\Omega}^* \to A$ is a DJFSC defined only over basis joint observations. A *basis hyperplane* is a vector of hyperplanes, one for each basis joint observation. We call the *complement hyperplane*, a vector of hyperplanes, one hyperplane for each non basis joint observation $o \in \Omega \backslash \mathring{\Omega}$. Any object that satisfies the decentralization constraint is said to be valid.

**Valid policies.** We introduce below a criterion that checks whether a DJFSC $\delta$ satisfies the decentralization constraint, *i.e.,* it is a *valid* policy.

LEMMA 1. *A DJFSC $\delta = (X, \pi, \eta, x_0)$ satisfies the decentralization constraint if and only if it corresponds to a vector of DFSCs $(\delta^1, \cdots, \delta^n)$ such that: $\forall x \in X, \exists (x^1, \cdots, x^n)$: $\pi(x) = (\pi(x^1), \cdots, \pi(x^n))$ and $\eta(x, o) = (\eta(x^1, o^1), \cdots, \eta(x^n, o^n))$ where $o = (o^1, \cdots, o^n)$ and $\delta^i = (X^i, \pi, \eta, x^i_0)$.*

This lemma states that the joint action taken for a given sequence of joint observations when the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ is executed is exactly the joint action taken when $\delta$ is executed after perceiving the same sequence of joint observations. Now we are ready to claim our main theorem.

THEOREM 1. *The basis DJFSC $\delta_B$ is the sufficient information to build a DJFSC $\delta$ that is valid.*

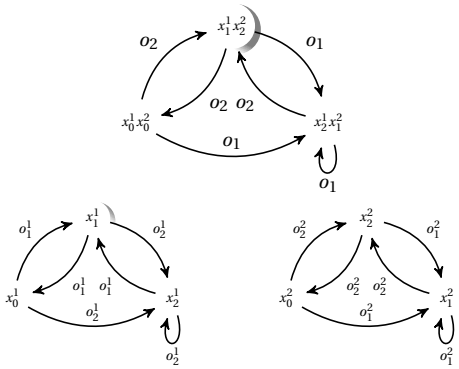

**Figure 1: Extraction of vector of DFSCs.**

To prove this we suggest a constructive two-step method that builds a unique DJFSC $\delta$ based on $\delta_B$: first, we build the unique vector of DFSCs associated with a basis $\delta_B$. To do so, we extract from $\delta_B$ the DFSC $\delta^i$ for each agent $i = 1 \cdots n$. This is achieved by removing from $\delta_B$ the components related to the other agents, such that only nodes and arcs that are labeled by individual actions and observations of agent $i$ are kept. This step provides us with the vector of DFSCs $(\delta^1, \cdots, \delta^n)$. We then build DJFSC $\delta$ based on the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ by using Lemma 1: $\forall x \in X, \pi(x) = (\pi(x^1), \cdots, \pi(x^n))$ and $\eta(x, o) = (\eta(x^1, o^1), \cdots, \eta(x^n, o^n))$, where $o = (o^1, \cdots, o^n)$ and $X = \otimes_{i=1}^n X^i$. One can merge together machine states $x \in X$ where the associated action $\pi(x)$ and the successor links $\eta(x, o)$ are the same. This two-step method proves the existence of such a DJFSC . But, we still need to prove that $\delta_B$ is the sufficient information to build $\delta$. This is achieved by removing either a node or an arc from $\delta_B$. In that case, the vector of DFSCs $(\delta^1, \cdots, \delta^n)$ will consist of DFSCs $\delta^i$ that lack a node or an arc. In accordance with this two-step method, we illustrate in Figure 1 the vector of DFSCs (down) extracted from a basis DJFSC (up) where the basis set is $\mathring{\Omega} = \{o_1 = (o^1_1, o^2_2), o_2 = (o^1_2, o^2_1)\}$.

**Valid hyperplanes.** A similar result for a basis hyperplane can be easily derived from the above theorem.

[1] Fictitious individual observations are added to agents if necessary such that the assumption always holds.

COROLLARY 1. *Let $\{\alpha^o\}_{o\in\mathring{\Omega}}$ be a basis hyperplane. There exists a unique complement hyperplane $\{\alpha^o\}_{o\in\Omega\setminus\mathring{\Omega}}$, such that hyperplane $\alpha=\sum_{o\in\Omega}\alpha^o$ is valid.*

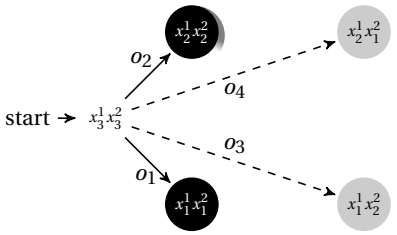


**Figure 2: The construction of a valid hyperplane (in white) and the complement of basis hyperplane hyperplane (in gray) given the basis hyperplane (in black).**

One can look at $\{\alpha^o\}_{o\in\mathring{\Omega}}$ as a set of hyperplanes that represents leaf nodes of a one-step basis joint policy $\delta_B$. Therefore, using Theorem 1, we are able to build the one-step joint policy $\delta$ associated to $\delta_B$ and thus the corresponding hyperplane. Consider the example depicted in Figure 2, where each agent has observation set $\Omega^i=\{o_1^i,o_2^i\}$ for $i=1,2$. Let $\mathring{\Omega}:=\{o_1=(o_1^1,o_2^2),o_2=(o_2^1,o_1^2)\}$ be the basis set, therefore its complement is $\Omega\setminus\mathring{\Omega}:=\{o_3=\{o_1^1,o_1^2\}$, $o_4=\{o_2^1,o_2^2\}\}$. We know $\alpha_{(x_3^1,x_3^2)}$ is built based on basis hyperplanes $\alpha_{(x_2^1,x_2^2)}$ and $\alpha_{(x_1^1,x_1^2)}$ that are associated to machine states $\eta(x,o_1)=(x_2^1,x_2^2)$ and $\eta(x,o_2)=(x_1^1,x_1^2)$, respectively. Thus, using Theorem 1, it turns out that the DFSC of agent 1 is transformed by adding machine state $x_3^1$ where $\eta(x_3^1,o_2^1)=x_2^1$ and $\eta(x_3^1,o_1^1)=x_1^1$, and the DFSC of agent 2 is transformed by adding machine state $x_3^2$ where $\eta(x_3^2,o_2^2)=x_2^2$ and $\eta(x_3^2,o_1^2)=x_2^1$. As a consequence, the complement hyperplane $\{\alpha_{\eta(x,o_3)},\alpha_{\eta(x,o_4)}\}$ is a vector of hyperplanes associated to machine states $(x_1^1,x_2^2)$ and $(x_2^1,x_1^2)$, respectively. For instance to determine the machine state associated to $\alpha_{\eta(x,o_3)}$, we use $\eta(x,o_3)=(\eta(x_3^1,o_1^1),\eta(x_3^2,o_1^2))$, *i.e.,* $(x_1^1,x_2^2)$. We call this procedure the construction of valid hyperplanes.

**Preserving the decentralization.** We are now ready to state the update and transformation rules that preserve the decentralization. It is straightforward to see that the construction of valid hyperplanes discussed above preserves the decentralization. Therefore, an update that preserves the decentralization consists in generating hyperplanes in $\Gamma_{\tau+1}$ using the construction of valid hyperplanes based on hyperplanes in the current set $\Gamma_\tau$. On the other hand, we identify two transformation rules that preserve the decentralization. The first rule is to remove individual machine states $x^i$ along with all joint machine states $x=x^ix^{-i}$ and hyperplanes $\alpha_x$ associated. The second rule is to replace an individual machine state $x^i$ by another one $\hat{x}^i$ every where $x^i$ appears. Because these are essentially transformations of individual controllers, the resulting joint controller is still valid.

## 3.2 The Sufficient Statistic

In this section, we explain how planning only over reachable beliefs allows the optimal policy of a DEC-POMDP to be found.

**Planning over reachable beliefs.** In order to be optimal, the Markov assumption requires that a policy depends on all the information available to the team at each time step. In DEC-POMDPs, at the execution time the agents are unaware of private information of their team-mates. However, in simulation each agent can divulge its private information to its team-mates. Therefore, the agents can maintain a complete joint history trace of all joint observations and joint actions they ever simulated. This joint history can get very long as time goes on. A well-known fact is that this joint history can be summarized via a belief. Unfortunately, because of the decentralization constraint a belief alone is not sufficient to condition the selection of a joint action. Nevertheless, we can still show that planning only over reachable beliefs allows the optimal policy to be computed. To better understand this, let $b_0$ be an initial belief we need to compute the optimal value. We know that the machine state $x$ whose hyperplane $\alpha_x$ yields the highest value for belief $b_0$ depends on machine states whose hyperplanes $\{\alpha_{\eta(x,o)}\}_{o\in\Omega}$ are selected for its successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$. Because of the decentralization constraint, the hyperplanes selected for successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$ are dependent on each other – that is the selection of hyperplanes for some successor beliefs in $\{b_{\pi(x),o}\}_{o\in\Omega}$ constrains the selection of hyperplanes for the remaining. As previously discussed, if we select a basis hyperplane for successor belief $b'\in\{b_{\pi(x),o}\}_{o\in\mathring{\Omega}}$ associated to basis joint observations, we determine directly the hyperplanes that are assigned to the remaining successor beliefs $\{b_{\pi(x),o}\}_{o\in\Omega\setminus\mathring{\Omega}}$. A similar argument can be used to show that successor beliefs of beliefs $\{b_{\pi(x),o}\}_{o\in\Omega}$ are also dependent on each other, and so on. Thus, in the decentralized multi-agent settings, the value of any belief $b_{\tau+1}$ depends on the value of all beliefs reachable starting in its precedent belief $b_\tau$. It is worth noting that because the initial belief $b_0$ does not have a precedent belief, its value depends only on all reachable beliefs starting in $b_0$. This permits us to claim that planning only over reachable beliefs allow us to compute the optimal policy for a given initial belief. Although, the optimal value $V_{\tau+1}(b)$ cannot be computed directly for each reachable belief (since it may depends on infinitely many other beliefs), the corresponding set $\Gamma_{\tau+1}$, that includes the hyperplane that is maximal for $b$ while satisfying the decentralization constraint, can be generated through a sequence of operations on the set $\Gamma_\tau$.

**Selection of belief set $B$.** Since the selection of a finite set of beliefs $B$ is crucial to the solution quality of all point-based algorithms, we rely on sampling techniques whose efficiency has been proven. In particular, [10] described a forward simulation that generates the sample belief set. The procedure starts by selecting the initial belief set $B_0:=\{b_0\}$ including the initial belief $b_0$ at time $\tau=0$. Then, for time $\tau=1,2,\cdots$, it expands $B_\tau$ to $B_{\tau+1}$ by adding all possible $b_{a,o}$ produced, and this $\forall b\in B_\tau$, $\forall a\in A$, $\forall o\in\Omega$, such that $pr(o|b,a)>0$. Then, the belief set $\bar{\Delta}:=\cup_{\tau=0}^{\infty}B_\tau$ is the set of beliefs reachable during the simulation time. It is therefore sufficient to plan only over these beliefs in order to find an optimal joint policy customized for a team of agents that starts from belief $b_0$, since $\bar{\Delta}$ constitutes a closed inter-transitioning belief set. Unfortunately, it is likely that $\bar{\Delta}$ is infinitely large. Thus, rather than keeping all possible $b^{a,o}$ we keep only a single $b^{a,*}$ that is the one that has the maximum $\mathbb{L}_1$ distance to the current $B$. And we add it into $B$ only if its $\mathbb{L}_1$ distance is beyond a given threshold $\varepsilon$.

# 4. POLICY ITERATION FOR DEC-POMDPS

Our PI ($\mathbb{H}_B$-PI) algorithm is summarized in Algorithm 1, with the principal structure shared with its POMDP counterparts [6, 8]. Similarly to the single-agent case, $\mathbb{H}_B$-PI consists of a threefold method: policy evaluation; policy improvement; and policy transformation. While the policy evaluation step is straightforward as previously discussed, processing the two later steps while providing guarantees on the satisfaction of the decentralization constraint is not trivial.

**Policy Improvement**. To show the importance of using basis

objects and sets for DP updates in decentralized multi-agent settings, we first consider state-of-the-art alternative strategies from either single-agent or multi-agent cases. In the single-agent case, we need first to generate the whole set $\Gamma_{\tau+1}$ and thereafter prune non valid hyperplanes [14]. This approach is, of course, hopelessly computationally intractable, as it requires the generation of $|A||\Gamma_\tau|^{\kappa^n}$ hyperplanes which is doubly exponential in $\kappa$ and $n$. In the multi-agent case, we build first all possible individual policies, that induces set $\Gamma_{\tau+1}$ of valid hyperplanes and thereafter prune dominated hyperplanes [2]. While this approach is more tractable than the previous one, the exhaustive backup limits its scalability. Indeed, set $\Gamma_{\tau+1}$ grows by $|A|\prod_{i=1}^{n}|X_\tau^i|^{|\Omega^i|}$ , which grows exponentially with $n$.

We are now ready to present the implementation of our backup operator $\mathbb{H}_B$. Similarly to the single-agent case, we start by creating intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$, $\forall a \in A$, $\forall o \in \mathring{\Omega}$ (*step* 1): $\Gamma^{a,o} \leftarrow \alpha_i^{a,o}(s) = \sum_{s'} P(s'|s,a)O(o|s',a)\alpha_i(s')$, $\forall \alpha_i \in \Gamma_\tau$ and $\Gamma^{a,*} \leftarrow \alpha^{a,*}(s) = R(s,a)$. Next we create $\Gamma^a$ ($\forall a \in A$), the cross-product over basis joint-observations, which includes one $\alpha^{a,o}$ from each $\Gamma^{a,o}$ (*step* 2): $\Gamma^a = \left(\otimes_{o\in\mathring{\Omega}} \Gamma^{a,o}\right) \otimes \Gamma^{a,*}$. Then, we create for each basis hyperplane $\alpha = \{\alpha^{a,o_1}, \cdots, \alpha^{a,o_\kappa}, \alpha^{a,*}\} \in \Gamma^a$ its complement hyperplane $\{\alpha^{a,o_{\kappa+1}}, \cdots, \alpha^{a,o_{|\Omega|}}\}$ with respect to Corollary 1 discussed above. We therefore replace in $\Gamma^a$, the hyperplane $\alpha$ by the hyperplane built as a cross-sum over hyperplanes in (*step* 3): $\forall \alpha \in \Gamma^a$, $\Gamma^a \leftarrow \Gamma^a \backslash \{\alpha\}$ and $\Gamma^a \leftarrow \alpha' = \alpha^{a,*} + \gamma \sum_o \alpha^{a,o}$. Afterwards, we take the union of $\Gamma^a$ sets (*step* 4): $\Gamma_{\tau+1} = \cup_a \Gamma^a$. Finally, we include the initial set of hyperplanes (*step* 5): $\Gamma_{\tau+1} = \Gamma_{\tau+1} \cup \Gamma_\tau$ to preserve the integrity of the solution. In practice, many of the hyperplanes $\alpha_i$ in the final set $\Gamma_{\tau+1}$ may be completely dominated by another hyperplane. To prune away those hyperplanes while preserving the decentralization, we rely on an individual machine state dominance criterion:

THEOREM 2. *Let $X^i$ and $X^{-i}$ be sets of machine states of agent $i$ and the other agents except agent $i$, respectively. A machine state $x^i \in X^i$ is dominated iff: $\exists \hat{x}^i \in X^i \backslash x^i : \alpha_{\hat{x}^i x^{-i}} \cdot b \geq \alpha_{x^i x^{-i}} \cdot b$, $\forall b \in B, \forall x^{-i} \in X^{-i}$.*

PROOF. We rely on a proof by contradiction. Assume (1) machine state $x^i \in X^i$ is non dominated and (2) $\exists \hat{x}^i \in X^i \backslash x^i : \alpha_{\hat{x}^i x^{-i}} \cdot b \geq \alpha_{x^i x^{-i}} \cdot b$, $\forall b \in B, \forall x^{-i} \in X^{-i}$. From the first claim, we derive that some of the reachable beliefs yield their optimal values under the decentralization constraint at hyperplanes $\alpha_{x^i x^{-i}}$, where $x^{-i} \in X^{-i}$. From the second argument, we derive that there exists a machine state $\hat{x}^i \in X^i$ such that hyperplane $\alpha_{\hat{x}^i x^{-i}}$ point-wise dominates hyperplane $\alpha_{x^i x^{-i}}$ for any machine state $x^{-i} \in X^{-i}$. In addition, by replacing machine state $x^i$ by machine state $\hat{x}^i$ the decentralization constraint still holds. As a result, this modification transforms the initial policy into a policy with a value function that increases for at least one belief state $b \in B$ and decreases for no $b \in B$ – which is a contradiction. □

The pruning then alternates from one agent to another until no more machine states are pruned. We extend the machine state dominance criterion to prune successively intermediate sets $\Gamma^{a,o}$ and $\Gamma^a$, for all $a \in A$ and $o \in \mathring{\Omega}$ and set $\Gamma_{\tau+1}$. As $\kappa$ and $|\Gamma_\tau|$ grow the overhead of performing these pruning mechanisms is non-negligible.

**The point-based B&B backup $\bar{\mathbb{H}}_B$.** This method aims at building the next set of hyperplanes $\Gamma_{\tau+1}$ using hyperplanes in $\Gamma_\tau$. The problem of finding the next set of hyperplanes $\Gamma_{\tau+1}$ given a finite set of beliefs $B$ and intermediate sets $\Gamma^{a,*}$ and $\Gamma^{a,o}$ ($\forall a \in A$ and $\forall o \in \mathring{\Omega}$), corresponds to the problem of finding basis hyperplane $(\alpha_b^{a,o_1}, \cdots, \alpha_b^{a,o_\kappa})$ such that the corresponding valid hyperplane $\alpha_b \in \Gamma_{\tau+1}$ is maximal for belief $b$, and this for each belief in $B$. One may suggest to solve such a problem using essentially the single-agent point-based backup operator while preserving the decentralization. That is, solving the problem by selecting hyperplane $\alpha_b^{a,o}$ that is maximal at $b$, one hyperplane for each basis joint observation. Then, we build the corresponding valid hyperplane. Unfortunately, the resulting hyperplane yields a lower-bound value. This is because its complement hyperplane can provide poor rewards. As a result, the overall contribution of the basis hyperplane is diminished by the poor contribution from its complement hyperplane. If we relax the problem by skipping the decentralization constraint, we build hyperplane $\alpha_b$ that is an upper-bound value at belief $b$ and potentially a non valid hyperplane. Hence, applying single-agent point-based methods do not lead to the best valid hyperplane for each belief. As this is essentially a combinatorial problem, we rely on a point-based branch-and-bound backup operator.

To describe this method, the following definitions are required: (a) $\vec{\alpha}^a$ is a vector of hyperplanes one for each joint-observation; (b) $\vec{\alpha}^a(o)$ is the selected hyperplane $\alpha_i^{a,o} \in \Gamma^{a,o}$. The forward search in the space of vectors of hyperplanes can be considered as an incremental construction of the best vector of hyperplanes based on optimistic evaluations of only partially completed vectors of hyperplanes. In each step of the search, the most promising partially completed vector of hyperplanes is selected and further developed, hence the best first approach. For a completely defined vector of hyperplanes $\vec{\alpha}^a$, we are able to find the corresponding hyperplane $\alpha = \alpha^{a,*} + \gamma \sum_o \vec{\alpha}^a(o)$. We then state our maximization problem as follows: $\alpha_b = \arg\max_\alpha (\alpha \cdot b)$. Notice that a partially completed vector of hyperplane is a vector where $\vec{\alpha}^a$ is only partially defined. Moreover, any partially completed vector of hyperplanes can be completed by assigning hyperplane $\alpha^{a,o} \in \Gamma^{a,o}$ that yields the highest value for belief $b$ at points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is not constrained. In order to determine whether or not to expand the leaf node of the search tree corresponding to a partially completed vector of hyperplanes, we compute an upper-bound value of any partially completed vector of hyperplanes for a given belief $b$. We define the upper-bound based on the decomposition of the exact estimate into two estimates. The first estimate, $G(\vec{\alpha}^a, b)$, is the exact estimate coming from points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is constrained. The second estimate, $H(\vec{\alpha}^a, b)$, is the upper-bound value coming from points $\vec{\alpha}^a(o)$ where $\vec{\alpha}^a$ is not constrained. We introduce sets of joint-observations $\Omega_1$ and $\Omega_2$ (such that $\Omega = \Omega_1 \cup \Omega_2$) that correspond to joint-observations that lead to constrained points and non-constrained points, respectively.

$$\bar{V}(\vec{\alpha}^a, b) = \underbrace{\left(\alpha^{a,*} + \gamma \sum_{o\in\Omega_1} \vec{\alpha}^a(o)\right) \cdot b}_{G(\vec{\alpha}^a,b)} + \underbrace{\left(\gamma \sum_{o\in\Omega_2} \max_{\alpha^{a,o}} (\alpha^{a,o} \cdot b)\right)}_{H(\vec{\alpha}^a,b)}$$

In the following, we draw our attention to a single search tree of our point-based B&B backup operator with the following entries: a belief $b$; a joint action $a$ and intermediate sets $\Gamma^{a,o}$ ($\forall o \in \Omega$), see Algorithm 2. We start by initializing the pool of *live* nodes with a partially defined vector $\vec{\alpha}_0^a$ where none of the points $\vec{\alpha}_0^a(o)$ ($\forall o \in \Omega$) is defined, and the value hereof is used as the value (called *incumbent*) of the current best solution (line 1). In each iteration of the heuristic, the node $\vec{\alpha}_i^a$ that yields the highest upper-bound is selected for exploration from the pool of live nodes (lines 3 – 4). Then, a branching is performed: two or more children of the node are constructed through the definition of a single point

$\vec{\alpha}_i^a(o)$ (line 5), for $o \in \mathring{\Omega}$. Furthermore, for each of the generated child nodes $\vec{\alpha}_i^a$, points $\vec{\alpha}_i^a(o)$ for $o \in \Omega \backslash \mathring{\Omega}$ that can be defined based on already constrained points are defined and the upper-bound is calculated. In case the node corresponds to a completely defined vector of hyperplanes its upper-bound is its exact estimate value, the value hereof is compared to the incumbent, and the best solution and its value are kept (lines 7 – 10). If its upper-bound is not better than the incumbent, the node is discarded, since no completely defined descendant nodes of that node can be better than the incumbent. Otherwise the possibility of a better solution in the descendant nodes cannot be ruled out, and the node is then joined to the pool of live nodes (line 11). When the search tree has been completely explored, the heuristic starts a new search tree with a new joint-action and the current best solution, until all joint-actions have been processed and this for each belief $b \in B$.

**Algorithm 2** Point-based branch and bound backup

```
1: procedure H̄_B-BACKUP(a, b, {Γ^{a,o}}_{o∈Ω})
2:     Initialize: Incumbent := V(b); Live := {α⃗_0^a}
3:     repeat
4:         Select α⃗_i^a ∈ Live : ∀α⃗_j^a ∈ Live, V̄(α⃗_j^a, b) ≤ V̄(α⃗_i^a, b)
5:         Live := Live \ {α⃗_i^a}
6:         Branch on α⃗_i^a generating α⃗_1^a, ··· , α⃗_{i_k}^a
7:         for 1 ≤ p ≤ k do
8:             if V̄(α⃗_{i_p}^a, b) > Incumbent then
9:                 if α⃗_{i_p}^a is completely defined then
10:                    Incumbent := V̄(α⃗_{i_p}^a, b)
11:                    Solution := α^{a,*} + Σ_o α⃗_{i_p}^a(o)
12:                else Live := Live ∪ {α⃗_{i_p}^a}
13:    until Live = ∅
14:    return Solution
```

**Policy Transformation.** [6] describes a policy transformation mechanism based on a simple comparison of $\Gamma_{\tau+1}$ and $\Gamma_\tau$, for the single-agent case. These rules are extended here for multiple agent settings. The procedure below iterates between agents until no more machine states can be eliminated. Notice that these transformation rules preserve the decentralization and decrease the value for no $b \in B$.

**Transformation rules**

For all $y^i \in X^i_{\tau+1}$:

- *rule* 1: If the action and successor links associated with $y^i$ duplicate those of a machine state of $\delta^i$, then keep that machine state unchanged in $\delta^i_{\tau+1}$;
- *rule* 2: Else if the machine $y^i$ dominates a machine state $x^i \in X^i_\tau$ associated with a machine state in $\delta^i_\tau$, change the action and successor links of that machine state to those that correspond to $x^i$;
- *rule* 3: Else add a machine state to $\delta^i_{\tau+1}$ that has the action and successor links associated with $y^i$;

Finally, prune any machine state for which there is no corresponding hyperplane in $\Gamma_{\tau+1}$, as long as it is not reachable from a machine state to which an hyperplane in $\Gamma_{\tau+1}$ does correspond. ♣

# 5. THEORETICAL ANALYSIS

This section presents convergence, error bound and the complexity arguments that draw on earlier approaches in single agent settings.

**Convergence Properties**. Point-based PI algorithms, despite being approximate methods, inherit many desirable properties of PI algorithms from POMDPs [6, 8], including: (1) If a DJFSC has not converged, the policy improvement transforms it into a DJFSC with a value function that increases for at least on belief $b \in B$ and decreases for no $b \in B$; (2) Point-based PI algorithms converge to an approximate DJFSC after a finite number of iterations; (3) the exact PI $\mathbb{H}$-PI algorithm ($B = \triangle S$) converges to an $\varepsilon$-optimal policy.

**Error bounds.** We now show that the error between $V_\tau$ and the optimal value function $V^*$ is bounded. The bound depends on how densely $B$ samples the belief set $\bar{\triangle}$; with denser sampling, $V_\tau$ converges to $V^*$. Cutting off the point-based PI ($\mathbb{H}_B$-PI) iterations at any sufficiently large time step, we know that the difference between $V_\tau$ and the optimal value function $V^*$ is not too large. The overall error in $\mathbb{H}_B$-PI, $\|V_\tau - V^*\|_\infty$, is bounded by: $\|V_\tau - V^*_\tau\|_\infty + \|V^*_\tau - V^*\|_\infty$. Because, $\mathbb{H}$ (resp. $\mathbb{H}_B$) is a contraction mapping, $\mathbb{H}V_\tau(b) = \max_a \mathbb{E}\left[R(a,b) + \gamma \sum_o V^*_\tau(b_{a,o})\right]$, the second term $\|V^*_\tau - V^*\|_\infty$ is bounded by $\gamma^\tau \|V^*_0 - V^*\|$ (see [5]). The remainder of this section states and proves a bound on the first term $\|V_\tau - V^*_\tau\|_\infty$. We prove that [10]'s bound stated for POMDPs holds for DEC-POMDPs. in the remainder of this paper, we state $\varepsilon_B = \max_{b' \in \bar{\triangle}} \min_{b \in B} \|b - b'\|_1$, and $\|r\|_\infty = \max_{s,a} R(s,a)$. First of all, let us prove the following lemma:

LEMMA 2. *The error introduced in $\mathbb{H}_B$-PI when performing one iteration of value function update over $B$, instead of $\bar{\triangle}$, is bounded by: $\eta \leq \frac{\|r\|_\infty}{1-\gamma} \varepsilon_B$*

PROOF. To have an intuition of the proof we provide an illustrative example Figure 3. Let $b' \in \bar{\triangle} \backslash B$ be the belief where $\mathbb{H}_B$-PI makes its worst error in the value function update, and $b \in B$ be the closest ($\mathbb{L}_1$ norm) sampled belief to $b'$. Let $\alpha_y$ the hyperplane that would be maximal at $b'$, where $y = \{y^i\}$ and $y \notin X$. Let $\alpha_x$ be the hyperplane that is maximal for $b'$, where $x = \{x^i\}$ and $x \in X$. By failing to include $\alpha_y$ in its solution set, $\mathbb{H}_B$-PI makes an error of at most $\alpha_y \cdot b' - \alpha_x \cdot b'$. On the other hand, we know that $\alpha_x \cdot b \geq \alpha_y \cdot b$. This is mainly because in order to remove hyperplane $\alpha_y$ Theorem 2 requires each machine state $y^i$ to be dominated or equal to machine state $x^i$ over belief set $B$. So,

$$\begin{array}{lll}
\eta & \leq & \alpha_y \cdot b' - \alpha_x \cdot b' \\
& = & \alpha_y \cdot b' - \alpha_x \cdot b' + (\alpha_y \cdot b - \alpha_y \cdot b) \quad \text{add zero} \\
& \leq & \alpha_y \cdot b' - \alpha_x \cdot b' + \alpha_x \cdot b - \alpha_y \cdot b \quad \alpha_x \text{ maximal at } b' \\
& = & (\alpha_y - \alpha_x) \cdot (b' - b) \quad \text{collect terms} \\
& \leq & \|\alpha_y - \alpha_x\|_\infty \|b' - b\|_1 \quad \text{Holder inequality} \\
& \leq & \|\alpha_y - \alpha_x\|_\infty \varepsilon_B \quad \text{definition of } \varepsilon_B \\
& \leq & \frac{\|r\|_\infty}{1-\gamma} \varepsilon_B \qquad \square
\end{array}$$

The remainder of this section states and proves a bound for $\mathbb{H}_B$-PI algorithm.

THEOREM 3. *For any belief set $B$ and any time step $\tau$, the error of the $\mathbb{H}_B$-PI algorithm $\eta_\tau = \|V_\tau - V^*_\tau\|_\infty$ is: $\eta_\tau \leq \frac{\|r\|_\infty}{(1-\gamma)^2} \varepsilon_B$.*

PROOF.

$$\begin{array}{lll}
\eta_\tau & = & \|V_\tau - V^*_\tau\|_\infty \quad \text{(definition of } \eta_\tau) \\
& = & \|\mathbb{H}_B V_{\tau-1} - \mathbb{H} V^*_{\tau-1}\| \quad \text{(definition of } \mathbb{H}_B \text{ and } \mathbb{H}) \\
& \leq & \|\mathbb{H}_B V_{\tau-1} - \mathbb{H} V_{\tau-1}\|_\infty + \|\mathbb{H} V_{\tau-1} - \mathbb{H} V^*_{\tau-1}\|_\infty \\
& \leq & \eta + \|\mathbb{H} V_{\tau-1} - \mathbb{H} V^*_{\tau-1}\|_\infty \quad \text{(definition of } \eta) \\
& \leq & \eta + \gamma \|V_{\tau-1} - V^*_{\tau-1}\|_\infty \quad \text{(contraction)} \\
& \leq & \eta + \gamma \eta_{\tau-1} \quad \text{(definition of } \eta_{\tau-1}) \\
& \leq & \frac{\|r\|_\infty}{(1-\gamma)^2} \varepsilon_B \quad \text{(series sum)} \quad \square
\end{array}$$

Notice that a tighter approximation error bound of $\mathbb{H}_B$-PI can be derived. Indeed, we estimate only the pruning error that incurs during the policy improvement step and ignore the policy

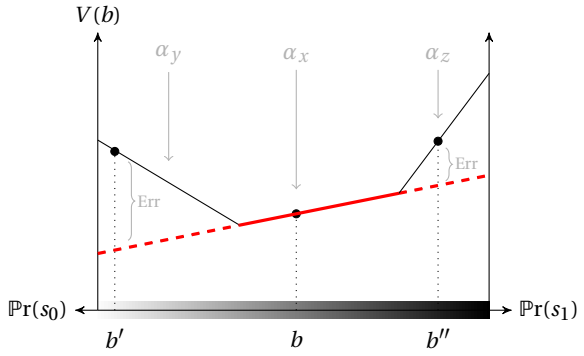


**Figure 3: Errors made on a 2-states DEC-POMDP by $\bar{\mathbb{H}}_B$-PI for beliefs $b'$ with respect to their closest belief $b$.**

evaluation step as well as hyperplanes from the earlier updates that are kept for the integrity of the controller.

## 6. EXPERIMENTAL RESULTS

As memery-bounded algorithms, NLP and BPI, are known to perform better than other approximate solvers, we compare $\bar{\mathbb{H}}_B$-PI. only to NLP and BPI. The comparison is made according to DEC-POMDP benchmarks from the literature: the multi-access broadcast channel (MABC) problem [7], the multi-agent tiger problem [9], the MEETING GRID problem [4], and the COOPERATIVE BOX-PUSHING problem [11]. The COOPERATIVE BOX-PUSHING problem provides an opportunity for testing the scalability of different algorithms. The reader can refer to the references above for an exact specification of the benchmarks. The performances of memory-bounded algorithms where sent by the authors, and run on an Intel(R) Pentium(R) 4 CPU 3.4GHz with 2GB of memory. Our algorithm was run on a Intel Core Duo 2.4GHz with 2GB of memory.

**Results.** Figure 4 presents our experimental results. For each problem and solver, we report: the value $V^{\delta}(b_0)$ at the initial belief $b_0$, the CPU time (in seconds) and the size $|B|$ belief set $B$ used for $\bar{\mathbb{H}}_B$-PI. We depict for each benchmark two graphs. On the right-hand side, we show the runtimes of all solvers over iterations. As performance results of memory-bounded solvers are built based on the solution size, we report on the $x$-axis the individual controller size. On the other hand, since $\bar{\mathbb{H}}_B$-PI is an iterative algorithm we report on the $x$-axis the number of iterations performed so far. As a result, we use iteration – that is the label of the $x$-axis, either for the size of individual controllers or the number of iterations performed so far, depending on the solver. Because all solvers do not have the same $x$-axis, the reader should mostly focus on the best value obtained by the solver and the amount of time it takes to reach that value. On the left-hand side, we illustrate the value at the initial belief over the iterations. Overall it appears that $\bar{\mathbb{H}}_B$-PI outperforms all NLP and BPI in all tested DEC-POMDP domains and in both computation time and solution quality. For problem of small-size, $\bar{\mathbb{H}}_B$-PI reaches a fixed point quite fast while neither NLP nor BPI were able to even reach the same solution quality for TIGER-A, MEETING-GRID, and COOPERATIVE BOX-PUSHING. As an example, $\bar{\mathbb{H}}_B$-PI takes 11.5 seconds to converge into a DJFSC of value 1.91 for the TIGER-A problem. The best memory-bounded solver, NLP, takes 1059 seconds to find its best joint stochastic controller of value −2.36. For this same problem, BPI could not find a stochastic joint controller of value higher than −52.63, this also explains why it does not appear in the graph. Even more importantly, $\bar{\mathbb{H}}_B$-PI converges into a fixed-point for MABC, TIGER-A, and MEETING-GRID benchmarks. This suggests that it inherited some of the properties of single-agent solvers.

## 7. CONCLUSION AND FUTURE WORK

We have derived and analyzed the first PI algorithms with provable error bounds. This research also provides interesting new theoretical insights, including the ability to perform DP updates while preserving the decentralization. Even though the experiment results demonstrate impressive improvement over the current standard methods, we are still far from being able to deal with real-world-size DEC-POMDPs. To this end, we are planning to use the new theoretical insights exhibited in this paper as foundations of more scalable algorithms facing some of the bottlenecks of our approach.

## 8. ADDITIONAL AUTHORS

## 9. REFERENCES


[1] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *UAI*, 2007.

[2] Daniel S. Bernstein, Christopher Amato, Eric A. Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of Markov decision processes. *JAIR*, 34:89–132, 2009.

[3] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4), 2002.

[4] Daniel S. Bernstein, Eric A. Hansen, and Shlomo Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, pages 1287–1292, 2005.

[5] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[6] Eric A. Hansen. An improved policy iteration algorithm for partially observable mdps. In *NIPS*, 1997.

[7] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715, 2004.

[8] Shihao Ji, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin. Point-based policy iteration. In *AAAI*, pages 1243–1249, 2007.

[9] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, pages 133–139, 2005.

[10] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, 2003.

[11] Sven Seuken and Shlomo Zilberstein. Improved Memory-Bounded Dynamic Programming for DEC-POMDPs. In *UAI*, 2007.

[12] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *JAAMAS*, 17(2):190–250, 2008.

[13] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable Markov decision processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[14] E. Sondik. The optimal control of partially observable Markov decision processes. Technical report, Standford, 1971.

[15] E. J. Sondik. The optimal control of partially observable Markov decision processes over the infinite horizon : Discounted cost. *Operations Research*, 12:282–304, 1978.

[16] Daniel Szer and François Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *ECML*, pages 389–399, 2005.

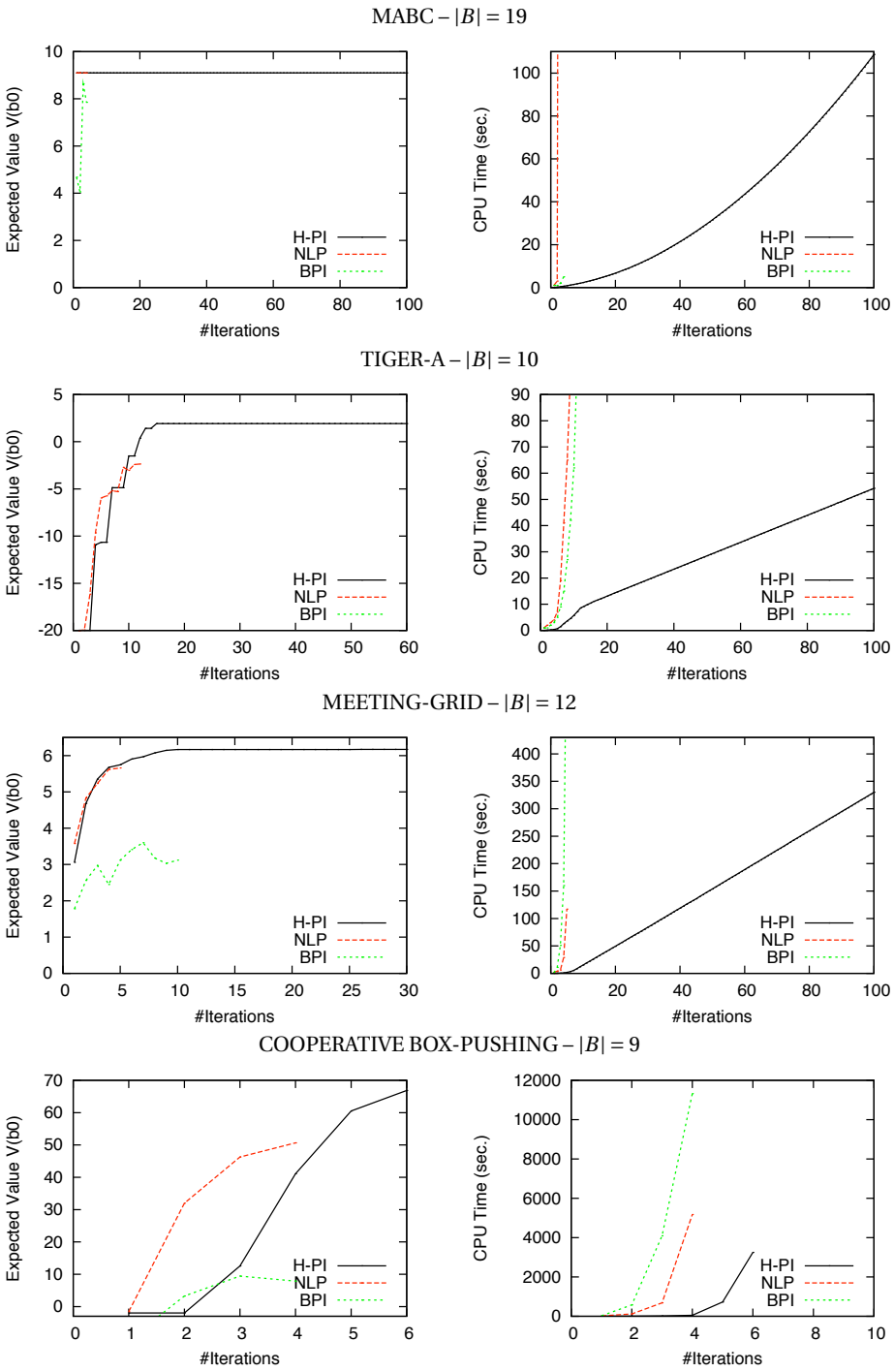


**Figure 4: Performance results for DEC-POMDP benchmark problems from the literature.**

# Two Level Recursive Reasoning by Humans Playing Sequential Fixed-Sum Games

Adam Goodie
Dept. of Psychology
University of Georgia
Athens, GA 30602
goodie@uga.edu

Prashant Doshi
Dept. of Computer Science
University of Georgia
Athens, GA 30602
pdoshi@cs.uga.edu

Diana Young
Dept. of Psychology
University of Georgia
Athens, GA 30602
dlyoung@uga.edu

## ABSTRACT

Recursive reasoning of the form *what do I think that you think that I think* (and so on) arises often while acting rationally in multiagent settings. Previous investigations indicate that humans do not tend to ascribe recursive thinking to others. Several multiagent decision-making frameworks such as RMM, I-POMDP and the theory of mind model recursive reasoning as integral to an agent's rational choice. Real-world application settings for multiagent decision making tend to be *mixed* involving humans and human-controlled agents. We investigate recursive reasoning exhibited by humans during strategic decision making. In a large experiment involving 162 participants, we studied the level of recursive reasoning generally displayed by humans while playing a sequential fixed-sum, two-player game. Our results show that subjects experiencing a strategic game made more competitive with fixed-sum payoffs and tangible incentives predominantly attributed first-level recursive thinking to opponents. They acted using second level of reasoning exceeding levels of reasoning observed previously.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

## General Terms

Theory, Performance

## Keywords

decision making, recursive reasoning, human behavior

## 1. INTRODUCTION

Strategic recursive reasoning of the form *what do I think that you think that I think* (and so on) arises naturally in multiagent settings. For example, a robotic uninhabited aerial vehicle (UAV)'s decision may differ if it believes that its reconnaissance target believes that it is not being spied upon in comparison to when the UAV believes that its target believes that it is under surveillance. Specifically, an agent's rational action in a two-agent game often depends on the action of the other agent, which, if the other is also rational, depends on the action of the subject agent.

Assumptions of *common knowledge* [11, 10] of elements of the game tend to preclude the emergence of recursive reasoning. However, not all elements can be made common knowledge. For example, an agent's belief is private especially in a non-cooperative setting. Multiple decision-making frameworks such as the recursive modeling method (RMM) [17, 18] and interactive partially observable Markov decision process (I-POMDP) [16, 8] model recursive beliefs as an integral aspect of agents' decision making in multiagent settings.

Real-world applications of decision making often involve mixed settings that are populated by humans and human-controlled agents. Examples of such applications include UAV reconnaissance in an urban operating theater and online negotiations involving humans. The optimality of an agent's decisions as prescribed by frameworks such as RMM and I-POMDP in these settings depends on how accurately the agent models the strategic reasoning of others. A key aspect of this modeling is the depth of the recursive reasoning that is displayed by human agents.

Initial investigations into ascertaining the depth of strategic reasoning of humans by Stahl and Wilson [25] and more recently, by Hedden and Zhang [21] and Ficici and Pfeffer [12] show that humans generally operate at only first or second level of recursive reasoning. Typically the first level, which attributes no recursive reasoning to others, is more prominent. Evidence of these shallow levels of reasoning is not surprising, as humans are limited by bounded rationality.

Increasing evidence in cognitive psychology [14, 15, 19] suggests that tasks which are ecologically more valid and which incorporate tangible incentives induce decisions in humans that are closer to being rational. Both these aspects were lacking in the experiments conducted by Hedden and Zhang [21]. We hypothesize that a strategic setting that is realistic, competitive and includes tangible incentives would increase participants' tendency to attribute levels of reasoning to others that reflect individuals' actual level of reasoning.

In this paper, we report on a large study that we conducted with human subjects to test our hypothesis. We constructed a task that resembled the two-player sequential game as used by Hedden and Zhang but made more competitive by incorporating fixed-sum outcomes and monetary incentives. Subjects played the game against a computer opponent, although they were led to believe that the opponent was human. Different groups of subjects were paired against an opponent that used no recursive reasoning (zero level) and opposite one that used first-level reasoning. We also manipulated the realism of the task between participants, with one group experiencing the task described abstractly while the other experienced a task that was structurally identical but described using a realistic cover story involving UAV reconnaissance.

Data collected on the decisions of the participants indicate that $(i)$ subjects acted accurately significantly more times when the opponent displayed first-level reasoning than when the opponent was

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

at zero level. The particpants also learned the reasoning level of opponents more quickly than reported previously by Hedden and Zhang. $(ii)$ No significant difference in the accuracy of the decisions was noticed between abstract and realistic task settings. Thus, our study reveals clear evidence that higher levels of recursive reasoning could be observed in humans under simpler and more competitive settings with tangible incentives. However, there is room for future research on whether increased realism contributes to deeper levels of strategic reasoning.

## 2. RELATED WORK

Harsanyi [20] recognized that indefinite recursive thinking arises naturally among rational players, which leads to difficulty in modeling it computationally. In order to, in part, avoid dealing with recursive reasoning, Harsanyi proposed the notion of agent types and common knowledge of the joint belief over the player types. However, as shown in [11, 10], common knowledge is itself modeled using an indefinite recursive system.

Since Harsanyi's introduction of abstract agent types, researchers have sought to mathematically define the type system. Beginning with Mertens and Zamir [24], who showed that a type could be defined as a hierarchical belief system with strong assumptions on the underlying probability space, subsequent work [5, 22] has gradually relaxed the assumptions required on the state space while simultaneously preserving the desired properties of the hierarchical belief systems. Along a similar vein, Aumann defined recursive beliefs using both a formal grammar [1] and probabilities [2] in an effort to formalize interactive epistemology.

Within the context of behavioral game theory [6], Stahl and Wilson [25] investigated the level of recursive thinking exhibited by humans. Stahl and Wilson found that only 2 out of 48 (4%) of their subjects attributed recursive reasoning to their opponents while playing 12 symmetric $3\times3$ matrix games. On the other hand, 34% of the subjects ascribed zero-level reasoning to others. Remaining subjects utilized either Nash equilibrium based or dominant strategies. Corroborating this evidence, Hedden and Zhang [21] in a study involving 70 subjects, found that subjects predominantly began with first-level reasoning. When pitted against first-level co-players, some began to gradually use second-level reasoning, although the percentage of such players remained generally low. Hedden and Zhang utilized a sequential, two-player, general-sum game, sometimes also called the Centipede game in the literature [4]. Ficici and Pfeffer [12] investigated whether human subjects displayed sophisticated strategic reasoning while playing 3-player, one-shot negotiation games. Although their subjects reasoned about others while negotiating, there was insufficient evidence to distinguish whether their level two models better fit the observed data than level one models.

Evidence of recursive reasoning in humans and investigations into the level of such reasoning is relevant to multiagent decision making in *mixed* settings. In particular, these results are directly applicable to computational frameworks such as RMM [17], I-POMDP [16] and cognitive ones such as theory of mind [9] that ascribe intentional models of behavior to other agents.

## 3. EXPERIMENTAL STUDY: HIGHER LEVEL RECURSIVE REASONING

In a large study involving human subjects we investigate whether subjects would generally exhibit a higher level of recursive reasoning under particular settings that are more typical of realistic applications.

We begin with a description of the problem setup followed by the participating population and our methodology for the experiment.

### 3.1 Problem Setting

In keeping with the tradition of experimental game research [7, 6] and the games used by Hedden and Zhang [21], we selected a two-player alternating-move game of complete and perfect information. In this sequential game, whose game tree is depicted in Fig. $1(a)$, player *I* (the leader) may elect to *move* or *stay*. If player *I* elects to move, player *II* (the follower) faces the choice of moving or staying, as well. An action of stay by any player terminates the game. Note that actions of all players are perfectly observable to each other. While the game may be extended to any number of moves, we terminate the game after two moves of player *I*.

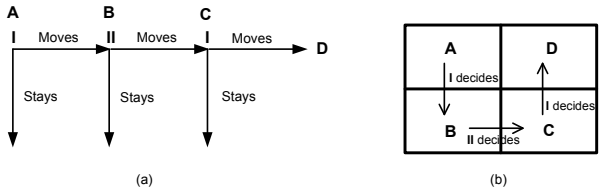


**Figure 1: $(a)$ A game tree representation (extensive form) of our two-player game. Because of its particular structure, such games are also sometimes called Centipede games. States of the game are indicated by the letters, A, B, C and D. $(b)$ Arrows denote the progression of play in the game. An action of *move* by each player causes a transition of the state of the game.**

In order to decide whether to move or stay at state **A**, a rational player *I* must reason about whether player *II* will choose to move or stay at **B**. A rational player *II*'s choice in turn depends on whether player *I* will move or stay at **C**. The default action is to stay. Thus, the game lends itself naturally to recursive reasoning and the level of reasoning is governed by the height of the game tree. Player *I*'s rational choice may be computed using backward induction in a straightforward way.

Recent studies have reported that more normative reasoning is facilitated in humans by using multimodal, spatial, ecologically valid (realistic) and experience-based presentations of the underlying game structure. Such facilitation has been reported in multiple psychological phenomena such as base-rate neglect [19, 15], overconfidence [14], the confirmation bias [13], and the conjunction fallacy [14]. In all these cases, the implementation of more realistic settings with richer contexts has made reasoning significantly closer to rational. Therefore, we imposed the following *cover story* in order to instantiate the game of Fig. 1:

*Player I (you) wants to gather critical information about a target. Player II (a UAV) wants to prevent you from gathering that information. In order to gather the information, you can use your best spy-grade binoculars, and you can move closer to the target. You are currently at position J in Fig. $2(a)$, where you are close enough to have a 60% chance of getting the information you need. If you move to position K, you will still have a 60% chance of getting the information. If you move to position L, you increase your chances to 100%. You cannot move directly from position J to position L, and you cannot move backwards (L to K, K to J, or L to J). Player II has equipment to jam your signal, which completely destroys any information you have obtained, if it is deployed successfully. Player II is currently at position X, and it can move to position Y, but it cannot move backwards (Y to X). If you stay at position J, you will not arouse Player II's suspicion, she will not attempt to jam your signal, and she will not move from position X.*

*If you move to K, then Player II must choose whether to stay at X or move to Y. If Player II stays at X, she has a 33.3% chance of jamming your signal, but if she moves to Y, she has a 66.7% chance of jamming your signal. However, if she chooses to move to Y, you can quickly move from K to L while she is moving. If you are at L and she is at Y, she has a 20% chance of jamming your signal. If you move from K to L while Player II is still at X, she has a 100% chance of jamming your signal, so you should not do that.*

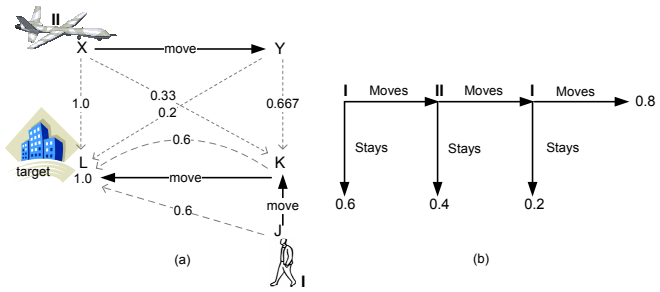


**Figure 2:** $(a)$ **A spatial visualization of the game where player** *I* **is a human intending to gather information about a target. Player** *II* **is a human-controlled UAV aiming to hinder** *I* **from gathering the critical information. The dashed arrows and probabilities indicate the chances of** *I* **gathering information or** *II* **hindering its access.** $(b)$ **Centipede representation of our game with the outcomes as the probabilities of success of player** *I***. It is a fixed-sum game and the remaining probability is the chance of success of player** *II* **(failure of player** *I***).**

This scenario is accompanied with Fig. 2 for illustration. If participants have difficulty in understanding the scenario, they will be further provided with a chart shown in Table 1 in which the posterior probabilities of success are clearly given. In order to succeed, player *I* must both obtain the information and not have the signal jammed. So the overall probability of success is: Pr(obtaining info)×[1-Pr(jamming signal)].

Notice from Fig. 2 that a rational player *I* will choose to stay. This is because if *I* chooses to move, player *II* will choose to stay with an overall chance of 0.6 of hindering access. A move by player *II* is not rational because player *I* will then choose to move as well with the probability of success for *II* being only 0.2.

| Player I's position | Player II's position | I's chance of obtaining info | II's chance of jamming signal | I's overall chance of success |
|---|---|---|---|---|
| J | X | 60% | 0% (doesn't try) | 60% |
| J | Y | wouldn't happen – II doesn't move if I doesn't go from J to K | | |
| K | X | 60% | 33.3% | 40% |
| K | Y | 60% | 66.7% | 20% |
| L | X | 100% | 100% | 0% (so don't try it!) |
| L | Y | 100% | 20% | 80% |

**Table 1: Chart showing the various probabilities for players** *I* **and** *II***.**

## 3.2 Participants

A total of 162 subjects participated in the study. The participants were undergraduate students enrolled in lower-level Psychology courses at the University of Georgia. In addition to receiving performance-contingent monetary incentives, which we describe below, the participants received partial course credit.

All participants gave informed consent for their participation prior to admission into the study. They were appropriately debriefed at the conclusion of the study.

## 3.3 Methodology

### 3.3.1 Opponent Models

In order to test different levels of recursive reasoning, we designed the computer opponent (player *II*) to play a game in two ways: $(i)$ If player *I* chooses to move, *II* decides on its action by simply choosing between the outcomes at states **B** and stay, and **C** with a default of stay in Fig. 1$(b)$ rationally. Therefore, *II* is a zero-level player and we call it *myopic* (see Fig. 3$(a)$). $(ii)$ If player *I* chooses to move, the opponent decides on its action by reasoning what player *I* will do rationally. Based on the action of *I*, player *II* will select an action that maximizes its outcomes. Thus, player *II* is a first-level player, and we call it *predictive* (see Fig. 3$(b)$).

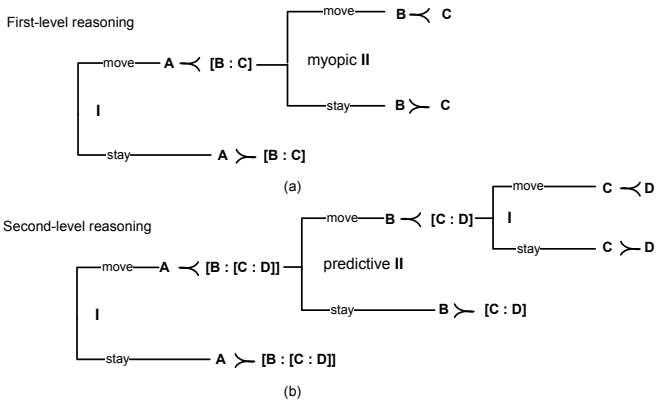


**Figure 3:** $(a)$ **A myopic player** *II* **decides on its action by comparing the payoff at state B with that at C. Here, B** $\prec$ **C denotes a preference of C over B for the player whose turn it is to play and B : C denotes the rational choice by the appropriate player between actions leading to states B and C. Thus, player** *I* **exhibits first-level reasoning.** $(b)$ **If player** *II* **is predictive, it reasons about** *I***'s actions. Player** *I* **then exhibits second-level reasoning in deciding its action at state A.**

To illustrate, in the game of Fig. 2 if player *I* decides to move, a myopic player *II* will move to obtain a probability of success of 0.8, while a predictive *II* will choose to stay because it thinks that player *I* will choose to move from **C** to **D**, if it moved. By choosing to stay, *II* will obtain an outcome of 0.6 in comparison to 0.2 if it moves.

### 3.3.2 Payoff Structure

Notice that the rational choice of players in the game of Fig. 1 depends on the preferential ordering of states of the game rather than actual values. Let $a \prec b$ indicate that the player whose turn it is to play prefers state $b$ over $a$, and because the game is purely competitive, the other player prefers state $a$ over $b$. Games that exhibit a preference ordering of **D** $\prec$ **C** $\prec$ **B** $\prec$ **A** and **A** $\prec$ **B** $\prec$ **C** $\prec$ **D** for player *I* are trivial because player *I* will always opt to stay in the former case and move in the latter case, regardless of how *II* plays. Furthermore, consider the ordering **C** $\prec$ **A** $\prec$ **B** $\prec$ **D** for player *I*. A myopic opponent will choose to move while a predictive opponent will stay. However, in both these cases player *I* will choose to move. Thus, games whose states display a preferential ordering of the type mentioned previously are not diagnostic – regardless of whether player *I* thinks that opponent is myopic or is predictive, *I* will select the same action precluding a diagnosis of

*I*'s level of recursive reasoning. Of all the 24 distinct preferential orderings among states that are possible, only one is diagnostic: **C** $\prec$ **B** $\prec$ **A** $\prec$ **D**. For this ordering, player *I* will move if it thinks that the opponent is myopic, otherwise *I* will stay if the opponent is thought to be predictive. We point out that the game in Fig. 2 follows this preference ordering.

### *3.3.3 Design of Task*

Batches of participants played the game on computer terminals with each batch having an even number of players. Each batch was divided into two groups and members of the two groups were sent to different rooms. This was done to create the illusion that each subject was playing against another, although the opponent was in reality a computer program. This deception was revealed to the subjects during debriefing.

Each subject experienced an initial *training phase* of at least 15 games that were trivial or those in which a myopic or predictive opponent behaved identically. These games served to acquaint the participants with the rules and goal of the task without unduly biasing them about the behavior of the opponent. Therefore, these games have no effect on the initial model of the opponent that participants may have. Participants who failed to choose the rational actions in any of the previous 5 games after the 15-game training phase continued with new training games until they met the criterion of no rationality errors in the 5 most recent games. Those who failed to meet this criterion after 40 total training games did not advance to the test phase, and were removed from the study.

In the *test phase*, each subject experienced 40 games instantiated with outcome probabilities that exhibited the diagnostic preferential ordering of **C** $\prec$ **B** $\prec$ **A** $\prec$ **D** for player *I*. The 40 critical games were divided into 4 blocks of 10 games each. In order to avoid subjects developing a mental set, we interspersed these games with 40 that exhibited the orderings, **C** $\prec$ **A** $\prec$ **B** $\prec$ **D** and **D** $\prec$ **B** $\prec$ **A** $\prec$ **C**. The latter games not only serve to distract the participants but also function as "catch" trials allowing us to identify participants who may not be attending to the games.

Approximately half the participants played against myopic opponents while the remaining played against predictive ones. In each category, approximately half of the participants were presented with just the Centipede representation of the games with probabilistic payoffs and no cover story, which we label as the *abstract* version. Remaining participants in the category were presented with the UAV cover story and the associated picture in Fig. 2, including the Centipede representation. We label this as the *realistic* version. About half of all participants also experienced a screen asking them what they thought the opponent would play and their confidence in the prediction, for some of the games.

Participants received a monetary incentive of 50 cents for every correct action that they chose in a game. This resulted in an average payout of approximately $30 per participant.

## 3.4 Results and Discussion

Our study spanned a period of *three months* from September through November 2008. We report the results of this study below.

### *3.4.1 Training Phase*

As mentioned before, each of the 162 human subjects initially played a series of 15 games in order to get acquainted with the fixed-sum and complete information structure, and objectives of the task at hand. After this initial phase, participants who continued to exhibit errors in any of the games up to 40 total games were eliminated. 26 participants did not progress further in the study. These participants either failed to understand how the game is played or exhibited excessive irrational behavior, which would have affected the validity of the results of this study.

### *3.4.2 Test Phase*

Of the 136 participants (70 female) who completed the test phase, we show the numbers that experienced myopic or predictive opponents and abstract or realistic versions of the games, in Table 2.

| Structure | myopic abstract | myopic realistic | predictive abstract | predictive realistic |
|---|---|---|---|---|
| No. of subjects | **37** | **30** | **37** | **32** |

**Table 2: The numbers of participants that experienced each of the 4 different types of tasks. The numbers differ from each other because of eliminations in the training phase.**

Participants in each of the 4 groups were presented with 40 instances of the particular game type whose payoff structure is diagnostic. For the sake of analysis, we assembled 4 *test blocks* each comprising 10 games. For each participant, we measured the fraction of times that the subject played accurately in each test block. We define an *accurate choice* as the action choice which is rational given the type of opponent. For example, in the game of Fig. 2, the accurate choice for player *I*, if the opponent is myopic, is to move. On the other hand, if the opponent is predictive, the accurate choice for *I* is to stay.

Because opponents types are fixed and participants experience 40 games, they have the opportunity to learn how their opponent might be playing the games. Consequently, participants may gradually make more accurate choices over time. Participants were deemed to have learnt the opponent's model at the game beyond which performance was always statistically significantly better than chance, as measured by a binomial test at the 0.05 level and one-tailed. This implies making no more than one inaccurate choice in any block of 10 games. (For this purpose, blocks were defined by a moving window of 10 games, not the fixed blocks used in other analyses.)

In Fig. 4$(a)$, we show the mean proportion of accurate choices across all participants in each of the 4 groups. Two group-level findings are evident from the results in Fig. 4$(a)$: First, the mean proportion of accurate choices is significantly higher when the opponent is predictive as compared to when it is myopic. This is further evident from Fig. 4$(b)$ where we show the mean proportions *marginalized* over the abstract and realistic versions of the tasks. Student t-tests with p-values $< 0.0001$ confirm that participants playing against predictive opponents have statistically significant higher proportions of accuracy compared to myopic opponents across all test blocks.

The higher proportions of accurate choice when the opponent is predictive in conjunction with the lower proportions when the opponent is myopic implies that subjects predominantly displayed second-level reasoning when acting. They expected the opponent to reason about their subsequent play (first-level reasoning) and acted accordingly. The fact that myopic opponents did not do this resulted in their choices being inaccurate.

Second, no significant difference in the mean proportions between abstract and realistic versions of the tasks is evident across any test block from Fig. 4$(a)$. This is regardless of whether the opponent is myopic or predictive. This observation is further evident in Fig. 4$(c)$ which shows the mean proportions for abstract and realistic versions marginalized over myopic and predictive opponents. Student t-tests with very low p-values revealed no statistical signif-

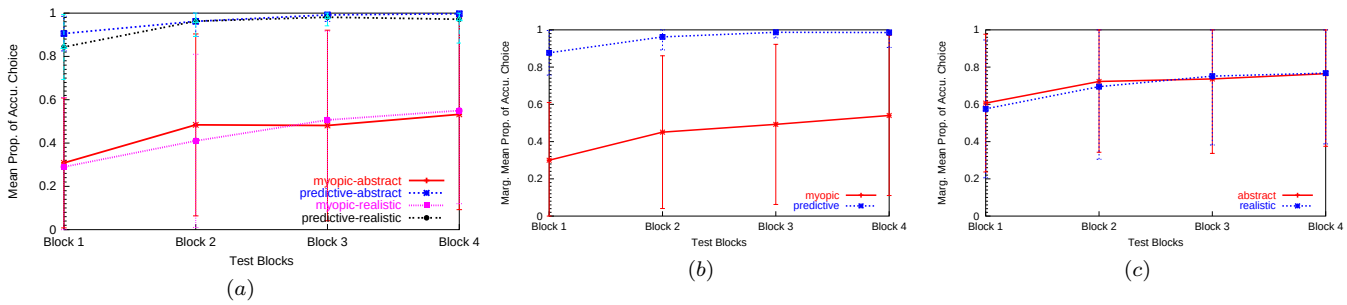


**Figure 4: $(a)$ Mean proportion of accurate choices of the participants for all conditions across test blocks. Notice that subjects generally expected their opponents to play at first level far more than at zero level. Mean proportions marginalized over $(b)$ abstract and realistic versions of the task, and $(c)$ over myopic and predictive opponents. The difference in the latter is not statistically significant.**

icance in the difference between the proportions, either overall or in any test block.

The lack of any significant difference in the accuracy of the choices seems to suggest that our cover story neither confounded the participants nor clarified it further in an intuitive sense. We speculate that the indifference is due to, $(i)$ the Centipede representation though abstract being sufficiently clear to facilitate understanding of this simple game, and thus $(ii)$ subjects playing the games with high accuracy leaving little room for improvement, at least in the predictive groups.

Notice from Fig. 4$(a)$ that the mean proportion of accurate choice improves over successive test blocks in all groups. Many participants in the predictive groups learnt in fewer than 10 games, by making no inaccurate choices in the first 7 games, and no more than one in the first 10 games. Forty-one participants, of which 40 were in the myopic conditions, never achieved learning by our standard; they were then assigned a value of 40 for the number of games to learning. The average number of games to learning was 10 for the predictive group and 31.7 for the myopic group (for the difference, Student t-tests reveal $p < 0.0001$).

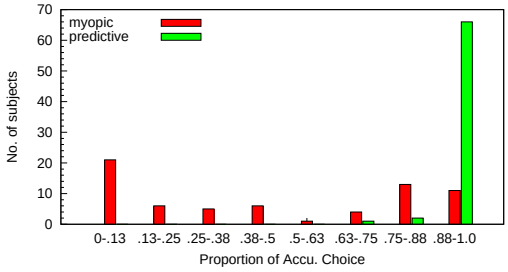


**Figure 5: Count of participants who played myopic or predictive opponents and grouped according to different proportions of accurate choice.**

Finally, in Fig. 5, we detail the number of participants whose actions across all games fell into different bins of proportion of accurate choice. Fig. 5 reveals that about 83% of the 67 participants who played against a myopic opponent had proportions less than 0.875. In comparison, only about 4% of the 69 participants who played a predictive opponent exhibited such proportions of accurate choice. Consequently, these results conclusively reveal that our subjects predominantly played reasonably and as though they expected the opponents to be predictive, and thus generally reasoned at a level higher than had been previously observed.

# 4. DISCUSSION

Multiple previous investigations into spontaneous recursive reasoning by individuals while playing strategic games have shown that humans *generally* do not think that opponents will think recursively while playing. Therefore, they themselves reason at only a single level of nesting. This has been attributed to the bounded rationality and limited mental processes of humans. Using a strategic game employed in one such previous study but made more competitive by incorporating fixed-sum payoffs and tangible incentives, we have shown that humans exhibit a higher level of recursive thinking when playing this game. Thus, we have demonstrated that in some settings humans *generally* reason at higher levels of recursion, and therefore exhibit more rational behavior. Consequently, our experiment opens up avenues for identifying settings where humans typically exhibit other forms of strategic sophistication such as simultaneous deeper and longer term thinking.

Although there is psychological evidence to suggest that intuitive cover stories induce human behavior closer to rational action, our cover story of UAV reconnaissance neither reduced nor improved the accuracy of the results. However, we do not claim that the previously perceived effect is not real, only that our particular cover story did not evoke it. We suspect that it may make a positive difference if the game is more sophisticated such as an extended Centipede game requiring three or four levels of recursive thinking.

An alternate hypothesis for explaining our results could be that many of the subjects solved the games completely using backward induction (ie. minimax). However, we do not believe this to be the case because, $(a)$ participants are implicitly encouraged to think about opponent models, $(b)$ Hedden and Zhang's result of predominant first-level reasoning in an identically-structured game precludes the use of backward induction. Others have noted that backward induction fails to explain a variety of human reasoning and decision-making behavior [23], and that it is not robust particularly in gaming situations where parity and certainty do not exist [3]. Our game does not display parity because player I has greater control over the outcome of the game, and $(c)$ the subjects were not explicitly trained in sophisticated game-theoretic techniques such as backward induction or minimax for solving games. Indeed, our exit questionnaire revealed that participants predominantly reasoned about the opponent's thinking.

# Acknowledgment

This research is supported in part by grant #FA9550-08-1-0429 from AFOSR.

## 5. REFERENCES


[1] R. J. Aumann. Interactive epistemology i: Knowledge. *International Journal of Game Theory*, 28:263–300, 1999.

[2] R. J. Aumann. Interactive epistemology ii: Probability. *International Journal of Game Theory*, 28:301–314, 1999.

[3] S. J. Brams and D. M. Kilgour. Backward induction is not robust: the parity problem and the uncertainty problem. *Theory and Decision*, 45:263–289, 1998.

[4] A. Brandenburger. The power of paradox: Some recent developments in interactive epistemology. *International Journal of Game Theory*, 35:465–492, 2007.

[5] A. Brandenburger and E. Dekel. Hierarchies of beliefs and common knowledge. *Journal of Economic Theory*, 59:189–198, 1993.

[6] C. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

[7] A. Colman. *Game Theory and Experimental Games*. Oxford: Pergamon Press, 1982.

[8] P. Doshi. On the role of interactive epistemology in multiagent planning. In *International Conference on AI and Pattern Recognition*, pages 208 – 213, 2007.

[9] R. Dunbar. Theory of mind and evolution of language. In *Approaches to the Evolution of Language*. Cambridge University Press, 1998.

[10] R. Fagin, J. Geanakoplos, J. Halpern, and M. Vardi. The hierarchical approach to modeling knowledge and common knowledge. *International Journal of Game Theory*, 28, 1999.

[11] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[12] S. Ficici and A. Pfeffer. Modeling how humans reason about others with partial information. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 315–322, 2008.

[13] L. Fiddick, L. Cosmides, and J. Tooby. No interpretation without representation: The role of domain-specific representations and inferences in the wason selection task. *Cognition*, 77:1–79, 2000.

[14] G. Gigerenzer. How to make cognitive illusions disappear: Beyond “heuristics and biases”. *European Review of Social Psychology*, 2:83–115, 1991.

[15] G. Gigerenzer and U. Hoffrage. How to improve bayesian reasoning without instructions: frequency formats. *Psychological Review*, 102:684–704, 1995.

[16] P. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multiagent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[17] P. J. Gmytrasiewicz and E. H. Durfee. A rigorous, operational formalization of recursive modeling. In *International Conference on Multi-Agent Systems*, pages 125–132, 1995.

[18] P. J. Gmytrasiewicz and E. H. Durfee. Rational interaction in multiagent environments: Coordination. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:319–350, 2000.

[19] A. Goodie and E. Fantino. Learning to commit or avoid the base-rate error. *Nature*, 380:247–249, 1996.

[20] J. C. Harsanyi. Games with incomplete information played by bayesian players. *Management Science*, 14(3):159–182, 1967.

[21] T. Hedden and J. Zhang. What do you think i think you think?: Strategic reasoning in matrix games. *Cognition*, 85:1–36, 2002.

[22] A. Heifetz and D. Samet. Topology-free typology of beliefs. *Journal of Economic Theory*, 82:324–341, 1998.

[23] R. D. McKelvey and T. R. Palfrey. An experimental study of the centipede game. *Econometrica*, 60:803–836, 1992.

[24] J. Mertens and S. Zamir. Formulation of bayesian analysis for games with incomplete information. *International Journal of Game Theory*, 14:1–29, 1985.

[25] D. Stahl and P. Wilson. On player’s models of other players: Theory and experimental evidence. *Games and Economic Behavior*, 10:218–254, 1995.

# Multiagent Coordination by Auctioning POMDP Tasks

Matthijs T.J. Spaan
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
mtjspaan@isr.ist.utl.pt

Nelson Gonçalves
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
ngoncalves@isr.ist.utl.pt

João Sequeira
Institute for Systems and Robotics
Instituto Superior Técnico
Lisbon, Portugal
jseq@isr.ist.utl.pt

## ABSTRACT

We present a procedure for the estimation of bids in auction protocols. This class of protocols is of interest to multiagent systems because they can be used to coordinate the assignment of tasks to agents. The main idea is to take advantage of methods for the synthesis of task execution controllers that rely on the availability of value functions. These provide a natural way to obtain the bid values for a given task. The approach is demonstrated on an active surveillance system, where mobile robots must approach and identify humans, and conduct patrols. The Partially Observable Markov Decision Process (POMDP) framework is used to compute policies for the execution of tasks by each agent, the task bid values are obtained directly from the respective value functions. Several simulation examples are presented for an urban surveillance environment, illustrating the applicability of our ideas.

## General Terms

Multiagent systems

## Keywords

POMDP, Auction Protocol, Task Assignment

## 1. INTRODUCTION

We present an approach to the estimation of bids in auction protocols. It is based in the value functions obtained from the design of controllers for the execution of tasks by the agents. From these functions, the fitness of an agent to execute a task (given the state of the environment) can be obtained directly, and without extra effort.

The use of auction protocols was initially proposed by [20] for collaborative, distributed problem solving among a set of agents. In multi-robot systems, these protocols are commonly used to determine the assignment of tasks [5]. They are also used in assignment problems arising in areas such as corporate management [2], and game theory [13]. The main advantages of these protocols are their robustness to individual agent failures and the reduced bandwidth requirements [6]. Another advantage is that the assignment solutions are computed in a distributed manner, and thus can be used by agents with low computational resources.

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

A crucial challenge to applying auction protocols is the estimation of the agents' bid values for each task. Agents must evaluate their fitness for executing a task using only locally available information. In mobile robotic applications, tasks often consist in the execution of a path [6, 7, 5]. Therefore, the bid value on each task can be the path distance, the travel time or a combination of these measures [12]. In general, these are heuristic measures that need to be defined for each task, often in an ad-hoc manner. By comparison, the advantage of using task controllers based on value functions is that bid functions need not be tailored for the application at hand. Instead, they already have been computed, and will reflect better true bid values, since they are derived directly from the task controller. In order to implement our task execution controllers using value functions, they are synthesized through a decision-theoretic approach. In particular, we use Partially Observable Markov Decision Processes (POMDPs) [10], which form a general and powerful mathematical basis for planning under uncertainty.

The remainder of this paper is as follows. In Section 2 we present an overview of the proposed approach. The POMDP framework is reviewed in Section 3, and Section 4 describes the auction protocol. In Section 5 the proposed approach for the estimation of bids is presented. Section 6 shows how the approach can be applied to an active surveillance system, and it is evaluated through simulation. Finally, in Section 7 we discuss the paper.

## 2. MULTIAGENT TASK COORDINATION

The problems considered in this paper are the assignment of tasks and their execution in a multiagent system. The first problem is formulated as the assignment of tasks with unknown arrival order. The agents can execute only one task at a time, which can be interrupted to begin the execution of another one. In this case, the current progress of the interrupted task is lost. Due to communication and hardware failures, the number of available agents is not known *a priori*. The computation of the assignment solution when the order of task arrivals is known and no failures occur is NP-HARD in general, [9, 4]. The available algorithms proposed for this problem often require computational resources organized in a centralized manner.

The second problem is the synthesis of controllers for the execution of each task by the agents. Each has available a finite, and possibly distinct, set of actions and can perceive features of interest in the environment. This problem is then formulated as computing a controller for the execution of a task by an agent. The tasks are assumed to

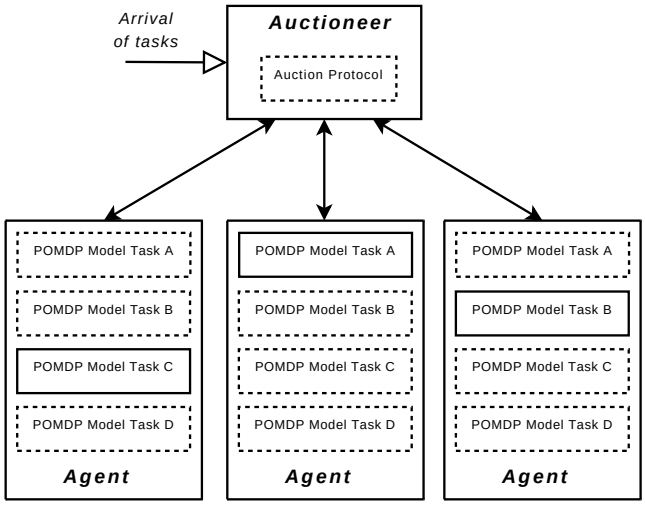


**Figure 1: Diagram of proposed solution. Each agent is running a POMDP model for each task in parallel, but only one is active (indicated by a solid box).**

be executed by a single agent, without explicit coordination with other agents. In this way, we avoid the severe complexity penalty involved when considering the full joint planning problem (either centralized or decentralized). Coordination is achieved on a task level, by finding the optimal assignment of individual tasks to agents.

A POMDP problem is formulated and solved for each task an agent can execute. The POMDPs at each agent receive the same set of local observations, but between agents no beliefs or other types of information are shared. When an agent is assigned a task, the policy of the corresponding POMDP is enabled and the others disabled. That is, the actions executed by the agent are those determined only by the policy of the assigned task POMDP.

The proposed approach in this paper is represented in the block diagram of Figure 2. It is composed by a central supervisor, denoted the *auctioneer*, and a set of heterogeneous agents. The tasks to be executed by the agent are received by the auctioneer and are then assigned through an auction protocol. Although the solution to the computation of task assignments is centralized, these can arrive in any order and the computational complexity is polynomial. Also, it is robust to communication and individual agent failures. The disadvantage is that in general, an optimal assignment solution is not guaranteed.

## 3. POMDP BACKGROUND

We will discuss POMDP models and solutions, briefly introducing some general background. A more elaborate POMDP model description is provided by [10], for instance.

A POMDP models the interaction of an agent with a stochastic and partially observable environment, and it provides a rich mathematical framework for acting optimally in such environments. The framework is based on the assumptions that at any time step the environment is in a state $s \in S$ and the action $a \in A$ is taken by the agent. As a result of this action, a reward $r(s,a)$ signal is received by the agent from the environment. And the environment state is changed to the new state $s'$, in accordance to a known stochastic transition model $p(s'|s,a)$. The task of an agent is defined by the reward it is given at each time step. The agent task goal is to maximize the long-term reward signals received. After the environment transition to the new state, an observation $o \in O$ is perceived by the agent. This is conditional on the current environment state, and possibly the action executed, according to a known stochastic observation model $p(o|s',a)$.

Given the transition and observation models, the POMDP can be transformed to a belief-state MDP, where the all past information of the agent is summarized using a belief vector $b(s)$. It represents a probability distribution over $S$, from which a Markovian signal can be derived for the planning of actions. The initial state of the system is drawn from the initial belief $b_0$, which is typically included in the POMDP problem formulation. Every time an action $a$ is taken by the agent and observation $o$ is obtained, the agent belief is updated by Bayes' rule; for the discrete case:

$$b_a^o(s') = \frac{p(o|s',a)}{p(o|a,b)} \sum_{s \in S} p(s'|s,a) b(s), \qquad (1)$$

where $p(o|a,b) = \sum_{s' \in S} p(o|s',a) \sum_{s \in S} p(s'|s,a) b(s)$, is a normalizing constant.

In POMDP literature, a plan is called a policy $\pi(b)$ and maps beliefs to actions. The policy can then be used to select the action the agent must execute in order to achieve the task goal. A policy $\pi$ can be characterized by a value function $V^\pi$ which is defined as the expected future discounted reward $V^\pi(b)$ the agent can gather by following $\pi$ starting from belief $b$:

$$V^\pi(b) = E_\pi \Big[ \sum_{t=0}^{h} \gamma^t r(b_t, \pi(b_t)) \Big| b_0 = b \Big], \qquad (2)$$

where $r(b_t, \pi(b_t)) = \sum_{s \in S} r(s, \pi(b_t)) b_t(s)$ following the POMDP model as defined before, $h$ is the planning horizon, and $\gamma$ is a discount rate, $0 \leq \gamma \leq 1$.

The process of solving POMDPs optimally is hard, and thus algorithms that compute approximate solutions are used. There has been much progress in approximate POMDP solving, see for instance [8, 21] and references therein. Furthermore, if a value function has been computed off-line, the on-line execution of the policy it implements does not require much computational requirements.

## 4. AUCTION PROTOCOL

The purpose of the auction protocol is to determine the POMDP policy that each agent must execute. This is equivalent, in the context of this paper, to the assignment of tasks to agents. The task generation process is assumed to be exogenous to the multiagent system. For instance, the execution of a task can be triggered by the occurrence of an event. The tasks arrive at the auctioneer at any time instant, but are assigned in a bulk manner at regular intervals. Note that we can also start a round of task assignment on demand, for instance when a high-priority task arrives. Also, a task can be scheduled to be executed at periodic time intervals, such as battery recharge operations.

The task execution requests could also originate from some of the agents or the auctioneer. As an example, the auction-

eer may directly receive event messages and locally favor the assignment of some tasks over others. The priority of each task is dictated by the specific application.

In order to solve the task assignment problem, the auctioneer is only required to know the expected discounted reward values of the POMDP task solutions from each agent, given their individual beliefs. The auction protocol is designed as follows, requesting this information from each agent.

DEFINITION 1 (AUCTION PROTOCOL). *The auction protocol is as follows:*

1) *All of the tasks are announced to the agents by the auctioneer.*
2) *The agents reply with their current expected discount reward $V^{\pi}(b)$ for each task. Hence, this is obtained from the solution $V^{\pi}$ for the task's POMDP model, and the agent's current belief $b$.*
3) *The assignment solution is computed by the auctioneer and announced to the agents.*

The assignment solution is determined by solving a mixed integer-linear program (MILP). The number of tasks waiting to be assigned by the auctioneer, at a particular instant, is represented by $w$. The number of agents that replied with bids is $n$. The assignment of the tasks to the agents is represented by the matrix $Y$. The element $y_{ij}$ is one if the $i$-th agent is assigned the $j$-th task, and zero otherwise. The optimal assignment solution, $Y^*$, is then determined from the MILP:

$$\begin{aligned} &\max \sum_i \sum_j \beta_{ij} \cdot y_{ij} \\ &s.t. \\ &\sum_i y_{ij} \leq 1, \quad j = 1, \ldots, w \\ &\sum_j y_{ij} \leq 1, \quad i = 1, \ldots, n \\ &y_{ij} \in \{0, 1\} \end{aligned} \tag{3}$$

where $\beta_{ij}$ is the value of the bid received from the $i$-th agent for the execution of the $j$-th task. The problem constraints state that each task is assigned to at most one agent and vice-verse. Thus, if their numbers are not equal, that is $w \neq n$, some tasks are not assigned or some agents remain idle. This can also occur in heterogeneous multiagent systems, where some agents may not be able to execute some of the tasks. In this case, the agents reply with an arbitrary negative value. From the problem formulation, it is clear that if $\beta_{ij}$ is negative then the $i$-th agent is not assigned the $j$-th task. This is because there is at least one feasible solution without this assignment and with a greater value for the cost functional.

The main advantage of this protocol is that the auctioneer is not required to known the number of available agents or their beliefs. Therefore, the approach is robust to the failure of agents or temporary network shortages. Because if an agent cannot be contacted, the others are still assigned tasks. Also, the communication and computational resources are reduced since only the current expected discounted reward must be reported to the auctioneer. Finally, the coordination of the agents is obtained implicitly through the auction protocol.

Although the assignment problem is a MILP, it can be efficiently solved in polynomial time using the Hungarian algorithm [3]. Therefore, this approach can be applied to small and medium sized problems with tens or hundreds of agents and tasks. In contrast, the auction protocols described in [5] often exhibit exponential complexity. The reason is that in these protocols, agents bid on bundles of tasks instead of the single task case of our protocol. Although the computational complexity is greatly reduced, the solution is only locally optimal. In [11] it was shown that for bundles with a small number of tasks, the assignment solution quality is improved without significantly increasing the computational and communication costs. Nevertheless the problem of computing the bid value is not considered and the tasks to be executed are known in advance. This is not the case in this paper, where the tasks and their arrival order are not known in advance. Also, it is assumed that agents do not accurately know their state. As a result, the estimation of bids for future tasks is complicated by the uncertainty at the current state.

## 5. POMDPS FOR BID ESTIMATION

In this work, we assume that the agents do not share any information among them. The reason is to reduce the network bandwidth and computational requirements, since the POMDP instances are smaller. It is known that relying on perfect communication can reduce the decentralized planning to a centralized one [17], but the size of the centralized problem still grows exponentially in the number of agents. Another reason is that since the agents are assumed not to be required to coordinate their individual actions in order to execute tasks, their POMDP models in general need not account for other agents. It must be stressed that the execution of tasks could benefit from knowledge on the other agents' beliefs and actions. For instance, if the planned paths of two mobile robots intersect, the collision could be avoided if their beliefs were shared. In order to avoid such potentially dangerous cases, low-level safety controllers are assumed to be available for the execution of actions.

Since multiple independent decision makers are present in the environment, the problem could be modeled as a decentralized POMDP (Dec-POMDP) [1, 19, 14]. However, given their very high complexity class, current algorithms do not scale to the types of applications we are focusing on. In our case, the coordination of the agents is obtained implicitly through the auction protocol and the auctioneer; coordination is considered on the level of task assignments vs. the level of individual agent actions, as is common in Dec-POMDPs.

A POMDP model of a certain task provides both an implementation of the task, i.e., how the agent should act to accomplish the task, as well as a valuation of the agent's fitness to execute it. The latter depends on the state of the environment, and in the POMDP case, on the agent's belief state. For each task, the reward model is such that when the agent accomplishes the goal, for instance reaching an area where to patrol, it receives a single reward of 10. When reaching the goal, the agent is transferred to an absorbing state, in which it receives zero reward, and it only leaves the absorbing state when a new task has been assigned. The use of an absorbing state is crucial, because otherwise the POMDP values can keep on rising, by instructing the agent to remain in a goal state. Although this effect might not

influence the policy implemented by the value function, inflated values are not desirable for our approach, given that we compare values between different POMDPs.

As discussed, in order to evaluate the relative benefit of using agent $x$ over agent $y$, their valuations should be in the same range. For this reason we employ the same maximum reward in each POMDP model, where the values of each are normalized to [0, 10]. However, we need to be able to express different priorities for tasks, in order to ensure that more important tasks get assigned first; for instance, when there are more tasks than agents available. This is accounted for by multiplying each the bid values by the task priority. Since the bid values are normalized, the result is that each bid is weighted by the respective task priority.

Note that instead of POMDPs also other planning models can be used, as long as they involve computing a value function. An example is the ALLIANCE control architecture [15], where the agent impatience and acquiescence levels determine the task to be executed. Assume that the sum of both levels is equal to some constant. Then the value function can be identified with the agent impatience and the acquiescence with the complementary value.

# 6. ACTIVE SURVEILLANCE SYSTEM

The presented approach is applied, in simulation, to an active surveillance system. It is composed by a set of mobile robots, an auctioneer and a network of cameras. These are capable of detecting, with some uncertainty, the location in the environment of robots and humans. Upon the detection of a human by the cameras, the auctioneer is notified. Note that here we present a simplified scenario, which can be extended easily to include more events (with different priorities), for instance the detection of fires.

## 6.1 Experimental setup

The robots have available on-board cameras, which can recognize humans, also with some uncertainty. Each robot can obtain its localization in the environment directly from the camera network. In addition, the robots' on-board power supply is limited and must be recharged after some time has elapsed. The tasks the mobile robots can execute are thus: (i) identification of humans, (ii) meeting a person, (iii) patrol the environment and (iv) recharge their on-board batteries. The first two tasks are assigned only when a person detection event has occurred. In these tasks the robot must approach the desired location and use the on-board sensors either to identify a human or meet it, in order to engage in human-robot interaction. The last two task types are assigned at regular intervals and have a low priority with respect to the first two. In this manner, if no events occur mobile robots can conduct patrols or recharge their batteries.

A set of four robots were simulated (as a unicycle), three modeled after a Pioneer 3-AT robot (indicated by Pioneer A, B and C), and one after an AtrvJr robot. In our current setup, the difference between the Pioneers and the AtrvJr is their maximum speed, which is $1.0\frac{m}{s}$ for the AtrvJr and $0.4\frac{m}{s}$ for the Pioneers. In addition, the camera of Pioneer A had a higher resolution than the cameras on-board the other robots. As a result, this robot could observe a location in the environment from a greater distance than the others.

A topological map of the active surveillance environment is represented in Figure 2. It was obtained from the test site of the URUS project [18], at the UPC campus in Barcelona,

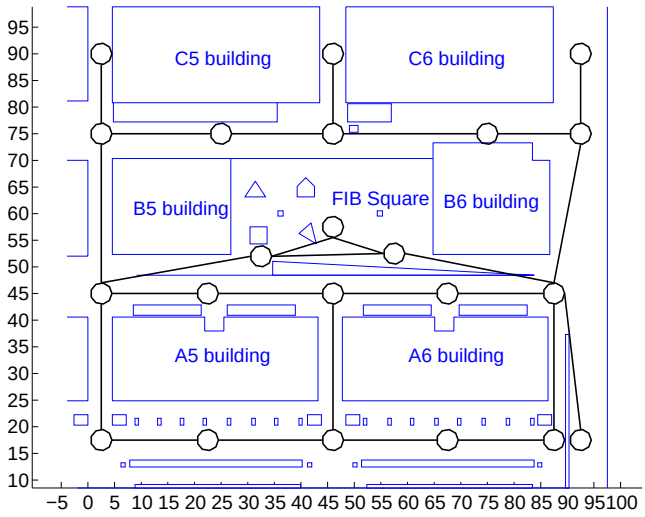


**Figure 2: Topological map of the active surveillance environment.**

| Task | State variables |
|---|---|
| Patrol SouthWest | Robot position |
| Patrol NorthWest | Robot position |
| Patrol NorthEast | Robot position |
| Patrol SouthEast | Robot position |
| Meet Person | Robot position, person position |
| Identify Person | Robot position, person position |
| Recharge | Robot position, battery level |

**Table 1: State variables used by different tasks. Positions are represented by nodes in the topological map; battery level consists of four levels, ranging from "high" to "very low".**

Spain. The overall dimensions of the map are 100 by 100 meters, as represented in the figure. The environment was partitioned in smaller regions, with the center of each represented in the map. The tasks are defined as navigation actions, defined using the region centers as way-points. In such a topological map, from each node a robot can only move to nodes connected to it by edges, representing the topology of the environment.

Each of the tasks mentioned in the previous section have been modeled and approximately solved a POMDP, using Symbolic Perseus [16]. The POMDP models are represented using two-stage dynamic Bayesian networks, and the software allows for exploiting (context-specific) independence between state variables. Table 1 lists the different state variables for each task. We assume the surveillance cameras can localize each robot, but with a particular uncertainty. Also each robot's movement actions are subject to noise. As we are essentially considering long-term plans, the discount rate is set high, $\gamma = 0.99$. Each movement action is penalized with a reward of $-0.1$.

## 6.2 Simulation Results

In a first experiment, all of the robots are initially positioned at the region containing the center node, in location $(46, 45)$, and are requested to execute four patrol tasks, one to each corner of the map. The value functions of each robot

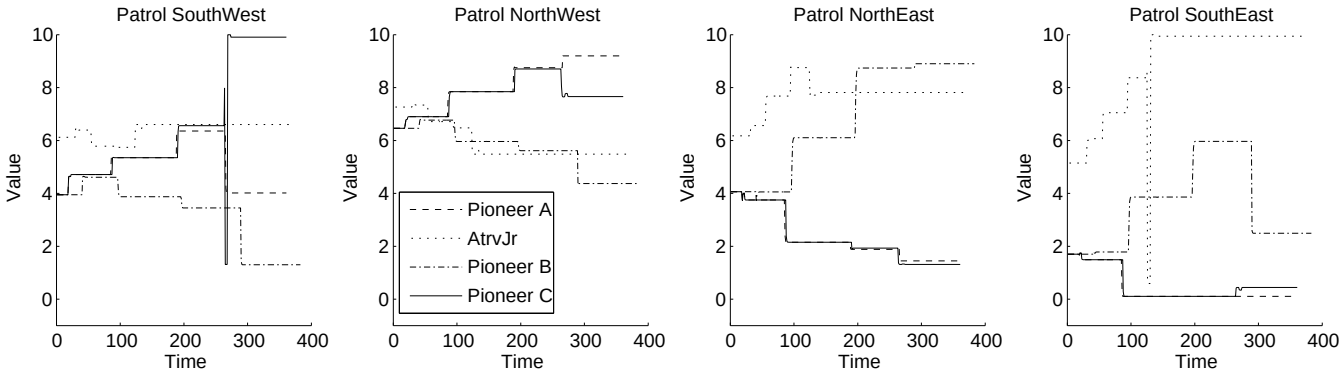


**Figure 3: Four different Patrol tasks.**

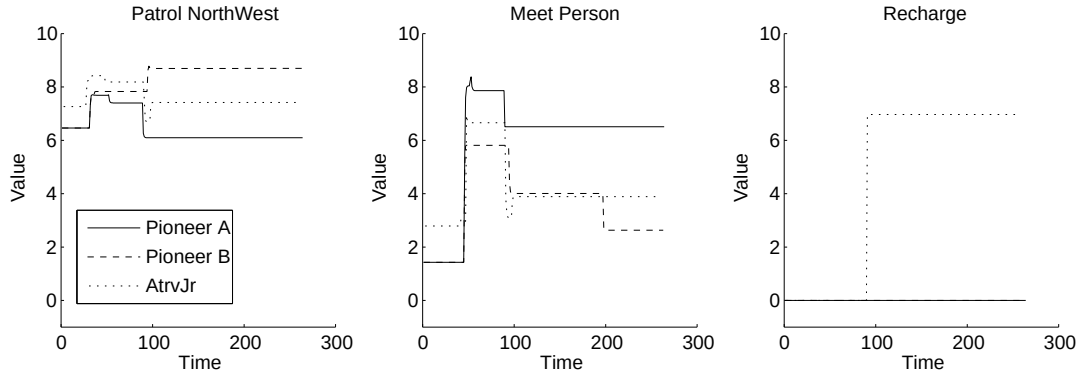


**Figure 4: Three Patrol tasks and a Meet person task.**

over time are plotted in Figure 3. As the belief of the robot changes while moving through the environment, the value for each of the tasks is updated. The AtrvJr robot has initially the highest value for any task since it is the fastest robot. Nevertheless it can only be assigned one task, in this case the "Patrol SouthEast" task, and the Pioneers are assigned the remaining three tasks: Pioneer A gets "Patrol NorthWest", Pioneer B "Patrol NorthEast", and Pioneer C "Patrol SouthWest". Although the AtrvJr robot has initially a higher value for the "Patrol NorthWest" task, it is assigned a different task. The reason is that the task assignment is determined by maximizing the sum of all bid values (3) and not individual bids.

Until about 100 time units, all Pioneers are still close to their initial starting position. As a result, their values for each task are similar, but a hysteresis mechanism prevents the assignment solution from changing too often. But as the robots move progressively away from the starting point, their values also become more different and the assignment solution stabilizes naturally.

In the second experiment (see Figure 4), three of the robots again start at the same node, but now at location $(46, 75)$. The robots are requested to execute three patrol tasks and also a recharge task. This is only executed when their battery level is low enough. Since the robots start with a full battery, initially each is assigned a patrol task. At about 50 time units, a person is detected in the node at location $(46, 90)$ and a task to meet the person is requested. Since the patrol and recharge tasks have a lower priority, one of the robots, Pioneer A, abandons its patrol task and was assigned the "Meet person" task. The other two robots are assigned two of the patrol tasks and one task is left unassigned. At around 100 time units, the AtrvJr robot, while moving to the patrol task goal, passes by the node containing the battery recharge station. Since the destination of the patrol task is still far and its battery is low, it is assigned the recharge task.

Finally, in the third experiment (Figure 5) robots Pioneer A and B are initially at nodes with locations $(2.5, 17.5)$ and $(87.5, 17.5)$ respectively. The robots where initially requested to execute two patrols tasks, one for each of their respective locations. Since the robots where already at their goals, they did not move. At about 40 time units, a person was detected at node $(46, 17.5)$ and an identify person task was requested. For this task, unlike the meet person, the robot is only required to approach the person close enough to take a clear picture. Although the robot Pioneer A is further away from the person, it has a camera with a better resolution and can take a picture at a greater distance. For this reason it is assigned the "Identify person" task instead of Pioneer B.

These experiments demonstrate that the auction protocol enable the robots to coordinate their task execution without communication of their state or beliefs. Also, the system is able to respond to detected events that occur during the execution of the tasks initially requested. Although in simulation, the presented methodology should transfer well to a real-world scenario, given the robustness with respect to communication failures.

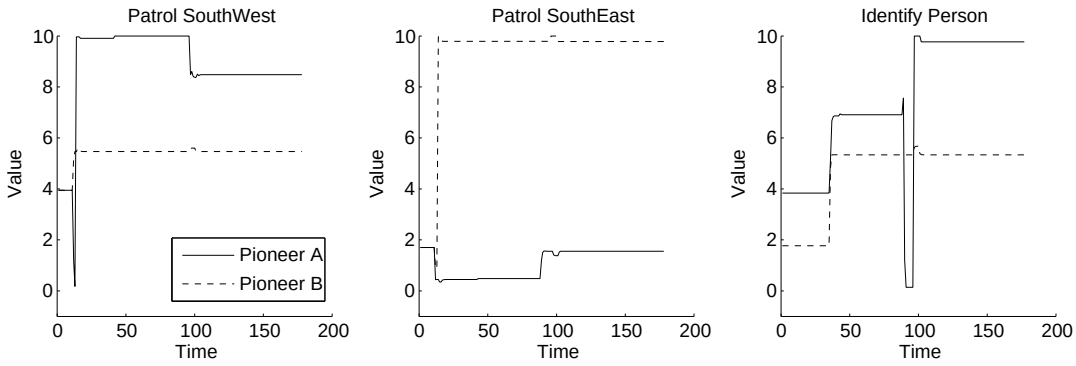


**Figure 5: Identify person task and two Patrol tasks.**

## 7. DISCUSSION AND FUTURE WORK

An approach to the assignment and execution of tasks in a multiagent system was presented. The motivation behind the approach was to illustrate the benefits of using auction protocols and the POMDP framework in multiagent systems. The auction protocols enable the coordination of multiple agents under stringent network operation conditions and robustness to individual agent failures. But an important drawback in the use of auctions is the question of how to estimate each agent's bid values.

The synthesis of controllers for the execution of tasks was performed using the POMDP framework. If suitable stochastic models of the environment and the agent observations are available, the synthesis problem can be formulated in a straightforward mathematical manner. The main difficulty of the POMDP approach is to compute a solution in an efficient manner. This is especially the case for almost all problem instances, expect those with relatively small dimensions.

The combination of the two frameworks produced a solution in which the individual drawbacks are mitigated. From the synthesis of controllers using POMDP task models, the values to bid are naturally obtained from the respective expected discounted rewards. Another advantage is that the agent's belief is already factored into this value. As a result, it is not necessary to invest additional time in the design of bid functions for each of the agents' tasks. Furthermore, as they are derived directly from the task controller, they are likely to reflect better true bid values, compared to commonly used heuristic bid functions.

The use of an auction enabled the use of smaller POMDP models than otherwise would be used if all agents and all tasks are considered simultaneously. This because the agents' coordination is implied in the use of the auction protocol and the auctioneer. Therefore, in the controller synthesis problem the other agents and tasks can be abstracted away. This is at the cost of optimality, since in practice the agents can interfere in each others' task execution.

It is possible, depending on the application, to use different auction protocols, such as on-line combinatorial auctions [13], or for agents to bid on bundles of tasks. This would require that the arrival order for tasks is known or that agents could accurately estimate their future rewards. Also, instead of POMDPs other planning models can be used, as long as they involve computing a value function.

A direction of future research is the synthesis of a controller for the auctioneer to determine the task priorities. The purpose is to maximize some performance criteria, such as the minimum assignment delay for some task types. The controller can also be used to determine which tasks to trade with other auctioneers.

## Acknowledgments

This work was supported by the European Project FP6-2005-IST-6-045062-URUS, ISR/IST pluriannual funding through the POS_Conhecimento Program that includes FEDER funds, and through FCT grant PTDC/EEA-ACR/73266/2006. Nelson Gonçalves is working under grant SFRH/BD/23804/2005 from Fundação para a Ciência e a Tecnologia.

## 8. REFERENCES

[1] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[2] M. Brydon. Economic metaphors for solving intrafirm allocation problems: What does a market buy us? *Decision Support Systems*, 42(3):1657–1672, 2006.

[3] R. E. Burkard. Selected topics on assignment problems. *Discrete Appl. Math.*, 123(1-3):257–302, 2002.

[4] M. I. Daoud and N. Kharma. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 68(4):399–409, April 2008.

[5] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[6] B. P. Gerkey and M. J. Mataric. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[7] B. P. Gerkey and M. J. Mataric. Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 3862–3868, 2003.

[8] M. Hauskrecht. Value function approximations for partially observable Markov decision processes.

*Journal of Artificial Intelligence Research*, 13:33–95, 2000.

[9] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner. A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers and Operations Research*, 36(2):358 – 378, 2009.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[11] S. Koenig and C. Tovey. Sequential bundle-bid singlesale auction algorithms for decentralized control. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 1359–1365, 2007.

[12] A. R. Mosteo and L. Montano. Comparative experiments on optimization criteria and algorithms for auction based multi-robot task allocation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3345–3350, 2007.

[13] N. Nisam, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, September 2007.

[14] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[15] L. E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[16] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, 2005.

[17] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[18] A. Sanfeliu and J. Andrade-Cetto. Ubiquitous networking robotics in urban settings. In *Workshop on Network Robot Systems. Toward Intelligent Robotic Systems Integrated with Environments. Procs. of 2006 IEEE/RSJ International Conference on Intelligence Robots and Systems (IROS2006)*, October 2006.

[19] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, Feb. 2008.

[20] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, 1980.

[21] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

# Exploiting Coordination Locales in Distributed POMDPs via Social Model Shaping

Jun-young Kwak*, Pradeep Varakantham+, Matthew Taylor*, Janusz Marecki++, Paul Scerri+, Milind Tambe*
*University of Southern California, Los Angeles, CA 90089, {junyounk, taylorm, tambe}@usc.edu
+Carnegie Mellon University, Pittsburgh, PA 15213, {pradeepv, pscerri}@cs.cmu.edu
++IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, marecki@us.ibm.com

## ABSTRACT

While distributed POMDPs provide an expressive framework for modeling multiagent collaboration problems, NEXP-Complete complexity hinders their scalability and application in real-world domains. This paper introduces a subclass of distributed POMDPs, and TREMOR, a novel algorithm to solve such distributed POMDPs. Two major novelties in TREMOR are (i) use of social model shaping to coordinate agents, (ii) harnessing efficient single agent-POMDP solvers. Experimental results demonstrate that TREMOR may provide solutions orders of magnitude faster than existing algorithms while achieving comparable, or even superior, solution quality.

## 1. INTRODUCTION

The excitement of Distributed Partially Observable Markov Decision Problems (DEC-POMDPs) flows from their ability to tackle real-world multi-agent collaborative planning, under transition and observation uncertainty [2, 3, 9, 17]. Given the NEXP-Complete complexity of DEC-POMDPs [3], however, the emerging consensus is to pursue approximate solutions [12, 17] and sacrifice expressivity by identifying useful subclasses of DEC-POMDPs (e.g., transition-independent DEC-MDPs [2, 15], and event-driven DEC-MDPs [1, 4, 10]). Such algorithms, through finding non-optimal joint policies or exploiting the structure of a subclass, are able to significantly reduce planning time.

In this continuing quest for efficiency, our research identifies a subclass of distributed POMDPs that allows for significant speedups in computing joint policies. We thus provide two key contributions. The first is a new subclass: *Distributed POMDPs with Coordination Locales* (DPCL). DPCL is motivated by the many domains, including those found in distributed POMDP literature, where multiple collaborative agents must perform multiple tasks. The agents can usually act independently, but they interact in certain *coordination locales*, identified as a set of states and times where agents could potentially need to coordinate, such as to avoid interfering with each other's task performance or to facilitate other agents' task performance. For example, in disaster rescue [8], multiple robots may act to save multiple injured civilians. While often acting independently, the robots should avoid colliding with other robots in a building's narrow corridors, and could clear up debris along the hallway to assist other robots. DPCL's expressivity allows it to model domains not captured in previous work: it does not require transition independence [2], nor does it require that agents' task allocation and coordination relationships be known in advance [1, 10], but does account for local observational uncertainty.

Our second contribution is a novel approach to solving DPCLs: *TREMOR* (Team's REshaping of MOdels for Rapid execution), an efficient algorithm for finding joint policies in DPCLs. TREMOR's primary novelty is that: (i) it plans for individual agents using single-agent POMDP solvers, thus harnessing the most efficient POMDP solution approaches; (ii) it then manages inter-agent coordination via *social model shaping* — changing the transition functions and reward functions of coordinating agents. While TREMOR is an approximate approach and it will not apply to the most general DEC-POMDPs, it does open a new line of attack on a large subclass of problems. We show that even in the presence of significant agent interactions, TREMOR can run orders of magnitude faster than state-of-the-art algorithms such as MBDP [17] and provides higher solution quality.

## 2. MOTIVATING DOMAINS

Our work is motivated by cooperative multiagent domains where agents must be assigned to different tasks. There are positive and negative interactions when agents perform these tasks [16, 20]. Since tasks are initially unassigned, agent interactions are initially unknown, but are limited to certain regions of the state space. Examples include disaster response [13] where fire-engines must be assigned to fight fires and ambulances to save civilians, wilderness search and rescue [5], and space exploration [6].

This paper focuses on urban disaster response where multiple robots must save civilians trapped in a building following a disaster. We use two types of robots, each of which must deal with sensing and action uncertainty. *Rescue* robots provide medical attention to victims. *Cleaner* robots remove potentially dangerous debris from building corridors, lobbies, and walkways. Saving victims provides a high reward, where the amount of reward depends on the victim's health status; cleaning up debris yields a lower reward (see Figure 1).

We model this as a discrete grid, where grid squares may be "safe" or "unsafe." Each agent begins with a health value of 3, which is reduced by 1 if it enters an unsafe square. An agent is disabled if its health falls to zero. Collisions may occur in narrow hallways if two robots try to pass through simultaneously, resulting in minor damage (cost) and causing one of the robots (chosen at random) to move back to its previous state. If a rescue robot attempts to traverse a "debris grid," it will get delayed by one time unit with high probability. A cleaner robot will instantly remove debris from a grid it is in and receive a small positive reward. [1]

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

[1]More details of the experimental domain and all DPCLs are shown in Appendix A & B.

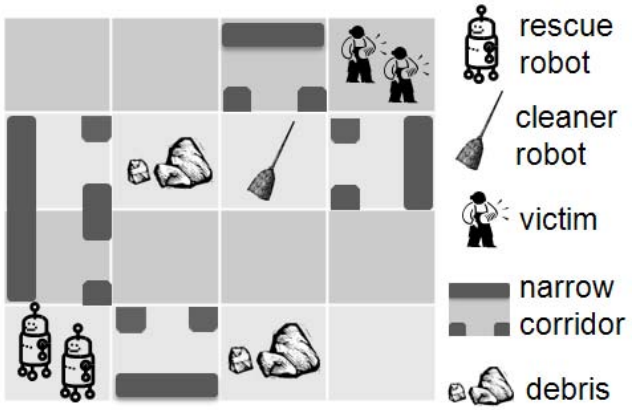


**Figure 1: This figure shows a $4 \times 4$ domain (with 1089 joint states). Two rescue robots plan to reach two victims. The rescue robots may collide in narrow corridors; a cleaner robot can remove debris to assist the rescue robots. Safeness of a grid cell (not shown in figure) is only known with a certain degree of certainty.**

Each agent has eight actions: move in the four cardinal directions and observe in each of the four cardinal directions. A movement action may succeed or fail, and observational uncertainty may lead to inaccurate information about movement success or safety of a location. Every action has a small cost and a rescue robot receives a high reward for being co-located with a victim, ending its involvement in the task. When modeling this domain as a DEC-POMDP, the goal of the planner is to obtain a reward-maximizing joint policy, where each policy assigns a rescue robot to a victim, and which debris (if any) each cleaner robot will clean.

# 3. THE DPCL MODEL

In a DPCL, a team of $N$ agents is required to perform a set of $M$ tasks, one agent per task but potentially many tasks per agent, in the presence of transitional and observational uncertainty. Like DEC-POMDPs, DPCL too is a tuple $\langle S, A, P, R, \Omega, O, b\rangle$ where $S$, $A$, and $\Omega$ and the sets of joint states, actions and observations; $P : S \times A \times S \rightarrow [0, 1]$, $R : S \times A \times S \rightarrow \Re$, and $O : S \times A \times \Omega \rightarrow [0, 1]$ are the joint transition, reward, and observation functions respectively and $b = \Delta S$ is a starting belief region. However, DPCL specializes from DEC-POMDPs in that it assumes $S := S_g \times S_1 \times \ldots \times S_N$ where $S_n$ is a set of local states of agent $n$ for $1 \leq n \leq N$ and $S_g = (E \times S_t)$ is a set of global states where $E = \{e_1, \ldots, e_H\}$ is the set of decision epochs and $S_t$ is a set of task states $s_t$ that keep track of the execution of tasks. Precisely, $s_t = (s_{t,m})_{1 \leq m \leq M}$ where $s_{t,m} \in \{$*Done, NotDone*$\}$ is the status of execution of task $m$.

Finding optimal joint policies to DEC-POMDPs is NEXP-Complete because the functions $P$, $R$ and $O$ are defined jointly, even if agent interactions are limited — DPCL is designed specifically to overcome this limitation. Let $\mathcal{P}_n : (S_g \times S_n) \times A_n \times (S_g \times S_n) \rightarrow [0, 1]$, $\mathcal{R}_n : (S_g \times S_n) \times A_n \times (S_g \times S_n) \rightarrow \Re$ and $\mathcal{O}_n : (S_g \times S_n) \times A_n \times \Omega_n \rightarrow [0, 1]$ denote agent local transition, reward and observation functions respectively. DPCL restricts DEC-POMDPs in that it assumes that agent observations are fully independent, i.e., $O((s_g, s_1, \ldots, s_N), (a_1, \ldots a_N), (\omega_1, \ldots \omega_N)) = \prod_{1 \leq n \leq N} \mathcal{O}_n((s_g, s_n), a_n, \omega_n)$ and that agent transitions and rewards are partially independent. Precisely, DPCL identifies situations where agent coordination is necessary, so that, with the exception of these situations, $P$ and $R$ naturally decompose into $\{\mathcal{P}_n\}_{1 \leq n \leq N}$ and $\{\mathcal{R}_n\}_{1 \leq n \leq N}$. These situations, referred to as *coordination locales* ($CL$s), are assumed in DPCL to be either same- or future-time. [2]

## 3.1 Same-time coordination locales (STCLs)

STCLs identify situations where state or reward resulting from the simultaneous execution of actions by a subset of agents cannot be described by the local transition and reward functions of these agents. Formally, a STCL for a group of agents $(n_k)_{k=1}^K$ is a tuple $cl_s = \langle (s_g, s_{n_1}, \ldots, s_{n_K}), (a_{n_1}, \ldots, a_{n_K})\rangle$ where $s_g$ is the current global state and $(a_{n_k})_{k=1}^K$ are the actions that agents $(n_k)_{k=1}^K$ execute in their current local states $(s_{n_k})_{k=1}^K$. For $cl_s$ to qualify as a STCL, there must exist joint states $s = (s_g, s_1, \ldots s_N), s' = (s'_g, s'_1, \ldots s'_N) \in S$ and a joint action $a = (a_n)_{n=1}^N \in A$ where $(s_{n_k})_{k=1}^K$ and $(a_{n_k})_{k=1}^K$ are specified in $cl_s$, such that the joint transition or reward function is non-decomposable, i.e., $P(s, a, s') \neq \prod_{1 \leq n \leq N} \mathcal{P}_n((s_g, s_n), a_n, (s'_g, s'_n))$ or $R(s, a, s') \neq \sum_{1 \leq n \leq N} \mathcal{R}_n((s_g, s_n), a_n, (s'_g, s'_n))$. The set of all STCLs is denoted as $CL_s$.

## 3.2 Future-time coordination locales (FTCLs)

FTCLs identify situations an action impacts actions in the future. Informally, because agents modify the current global state $s_g = (e, s_t)$ as they execute their tasks, they can have a future impact on agents' transitions and rewards since both $\mathcal{P}_n$ and $\mathcal{R}_n$ depend on $s_g$. Formally, a FTCL for a group of agents $\{n_k\}_{k=1}^K$ is a tuple $\langle m, (s_{n_k})_{k=1}^K, (a_{n_k})_{k=1}^K\rangle$ where $m$ is a task number and $(a_{n_k})_{k=1}^K$ are the actions that agents $(n_k)_{k=1}^K$ execute in their current local states $(s_{n_k})_{k=1}^K$. For $cl_f$ to qualify as a FTCL, the actual rewards or transitions of agents $(n_k)_{k=1}^K$ caused by the simultaneous execution of actions $(a_{n_k})_{k=1}^K$ from states states $(s_{n_k})_{k=1}^K$ must be different for $s_{t,m} =$ *Done* and *NotDone* for some global state $s_g = (e, s_t) \in S_g$. Precisely, there must exist: (i) starting joint states $s = (s_g, s_1, \ldots s_N), \overline{s} = (\overline{s}_g, s_1, \ldots s_N) \in S$ where $(s_{n_k})_{k=1}^K$ are specified in $cl_f$ and $s_g = (e, s_t)$ differs from $\overline{s}_g = (e, \overline{s}_t)$ only on $s_{t,m} \neq \overline{s}_{t,m}$; (ii) a joint action $a = (a_n)_{n=1}^N \in A$ where $(a_{n_k})_{k=1}^K$ are specified in $cl_f$ and (iii) ending joint states $s' = (s'_g, s'_1, \ldots s'_N), \overline{s}' = (\overline{s}'_g, s'_1, \ldots s'_N) \in S$ where $s'_g = (e', s'_t)$ differs from $\overline{s}'_g = (e', \overline{s}'_t)$ only on $s'_{t,m} \neq \overline{s}'_{t,m}$ such that either $P(s, a, s') \neq P(\overline{s}, a, \overline{s}')$ or $R(s, a, s') \neq R(\overline{s}, a, \overline{s}')$. The set of all FTCLs is denoted as $CL_f$.

*Example*: Consider a rescue robot from the domain in Section 2, entering a narrow corridor. If another robot were to attempt to enter the same narrow corridor simultaneously, one of them would transition back to starting state and the robots would damage each other (STCL). If the narrow corridor had debris and a cleaner robot completed the task of removing this debris, the rescue robot would traverse the corridor faster (FTCL).

# 4. SOLVING DPCLS WITH TREMOR

We are interested in providing scalable solutions to problems represented using the DPCL model. To this end, we provide TREMOR, an approximate algorithm that optimizes expected joint reward while exploiting coordination regions between agents. TREMOR accounts for the coordination locales, using a two stage algorithm: (1) A branch and bound technique to efficiently search through the space of possible task assignments. (2) Evaluating task assignments (for step (1) above) in the presence of uncertainty (transitional and observational) and coordination locales.

[2]If agent interactions are limited, $|CL_s| + |CL_f| \ll |dom(P)|$ and DPCLs are easier to specify than equivalent DEC-POMDPs.

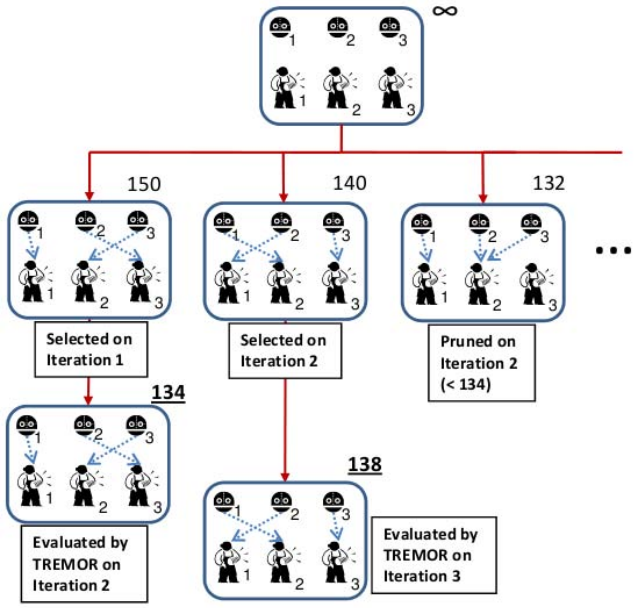


**Figure 2: This diagram depicts the branch and bound search.**

## 4.1 Branch and Bound Search

Multiagent planning problems often have a large number of possible task assignments, precluding exhaustive evaluation. TREMOR incorporates a *Breadth-first Branch and Bound* search algorithm to exploit task decomposition among a team, significantly pruning the search space. In order to aid the search, we compute upper bounds on the expected value of joint policy using a heuristic that solves the decision problems of agents as MDPs (ignoring the observational uncertainty). Search begins with computation of upper-bounds for all task assignments and evaluation of the task assignment with highest upper-bound using TREMOR. Any assignment with an upper-bound lower than a complete evaluation calculated by TREMOR is pruned. Task assignments [3] with the highest heuristic evaluations are repeatedly evaluated until all remaining allocations are evaluated or pruned (see Figure 2).

## 4.2 Task Assignment Evaluation

At this point of algorithmic execution, agents have been assigned their tasks and an optimal joint policy consistent with this assignment has to be found. Since the problem is still NEXP-Complete, TREMOR's approach in evaluating the current task assignment is to search for a locally optimal joint policy (see Algorithm 1). To that end, TREMOR initially finds the optimal joint policy assuming that agents are not interacting, i.e., by solving individual agent POMDPs (lines 1–3). Note that we can employ state of the art POMDP solvers to solve a POMDP in SOLVEPOMDP() (line 3). We then try to improve the joint policy (lines 5–41) until no agent policies can be changed.

At each iteration, we re-compute policies $\pi_i$ for all agents which are part of the sets $I_n$, where $1 \leq n \leq N$. This set includes agents whose local transition, $\mathcal{P}_i$, and reward functions, $\mathcal{R}_i$, have been changed due to interactions with agent $n$. TREMOR considers interactions due to STCLs (lines 6–21) and FTCLs (lines 22–38) separately.

[3]Note that TREMOR allows the number of agents and tasks to be unequal, as well as allowing an agent to be assigned to multiple tasks.

Upon verifying that a STCL $c$, $\langle (s_t, s_{n_1}, \ldots, s_{n_K}), (a_{n_1}, \ldots, a_{n_K}) \rangle$, involves agent $n$ (line 8), the algorithm computes the difference $R^+ - R^-$ in the expected utility, $EU(\pi_n)$ for agent $n$'s policy, given that the transition, $\mathcal{P}_n$ and reward functions, $\mathcal{R}_n$ of agent $n$ are updated for state action pairs in $c$. TREMOR then computes the probability, $\hat{c}$, that $c$ will occur given the current joint policy, $\pi$, and uses $\hat{c}$ to determine the *shaping reward* $R^\Delta$. Depending on whether $c$ is beneficial to agent $n$ or not, the algorithm behaves differently.

If the shaping reward is positive (beneficial to agent $n$; lines 15–17), agents are encouraged to follow policies that induce $c$. The agent is influenced by adding a fraction $R^\Delta/K$ of the shaping reward to local reward function $\mathcal{R}_i$ of each agent. To ensure a coherent dynamic model for the agents after interaction, local transition models of agents are then redefined by using the global transition function $P$ (for local state-action pairs resulting in $c$); such redefinition could potentially take into account the probability of coordination locales (although not used in our implementation). To calculate the old transition functions of each agent, $P_i$, we first "extract" the old probability of transitioning from one agent state to another given its action and a status of task, $s_t$. Let $e, e' \in E$ be the starting and ending decision epochs for that transition, $a_i \in A_i$ be the agent's action, $s_i, s_i' \in S_i$ be the starting and ending local agent states, $s_t, s_t' \in S_t$ be the starting and ending task states. The local transition probability of agent $i$ assuming a STCL $c$ does not occur, $P_{i,\neg c}(((e, s_t), s_i), a_i, ((e', s_t'), s_i'))$, is given as a domain input. We derive the local transition probability of agent $i$ assuming $c$ occurs, $P_{i,c}(((e, s_t), s_i), a_i, ((e', s_t'), s_i'))$, from a given joint transition function, and update $P_i$ using $\hat{c}$ and derived probabilities. Formally:

$$P_{i,c}(((e, s_t), s_i), a_i, ((e', s_t'), s_i')) \leftarrow$$

$$\sum_{\left\{\forall s'_{n_k} \in S, k \neq i\right\}} P((s_{n_1}, \ldots, s_{n_K}), (a_{n_1}, \ldots, a_{n_K}), (s'_{n_1}, \ldots, s'_{n_i}, \ldots, s'_{n_K})),$$

$$P_i \leftarrow \hat{c} \times P_{i,c}(((e, s_t), s_i), a_i, ((e', s_t'), s_i'))$$

$$+(1 - \hat{c}) \times P_{i,\neg c}(((e, s_t), s_i), a_i, ((e', s_t'), s_i'))$$

In contrast, if the shaping reward is negative (not beneficial to agent $n$; lines 18–21) agents in coordination locale $c$ are discouraged from policies that induce $c$, except for agent $n$ which is given no incentive to modify its behavior. As $c$ will not occur in this interaction, there is no need to redefine the agent local transition functions in terms of the joint transition function $P$. To update the reward and transition functions in an STCL, consider the following example from our domain. A STCL occurs when two robots, $i$ and $j$, bump into each other in a narrow corridor. We are initially given the transition probability of an individual robot $i$'s traveling through the corridor alone (as input). When two robots' policies create an STCL (agents $i$ and $j$ bump into each other in the narrow corridor), we first check if the STCL is beneficial or not. If it is non-beneficial, we provide a negative reward to one of the robots (robot $j$) to encourage it to avoid the narrow corridor; the robots' transition functions are not modified since this STCL will not occur. Although this would not happen in our example domain, a beneficial STCL would need a positive shaping reward. We then update the transition function $P_i$ of robot $i$, using the transition probabilities $P_i$ when bumping occurs and when it does not occur, using the updating formula above.

TREMOR then considers all FTCLs:
$c \in CL_f$, $\langle m, (s_{n_k})_{k=1}^K, (a_{n_k})_{k=1}^K \rangle$, involving agent $n$ (lines 22–38). To that end, it computes probabilities, $P_\pi^{e, s^+_{t,m}}$, that a task

is completed by a decision epoch, $e$, when the joint policy $\pi$ is executed. These probabilities are used to determine the sum of expected utilities for the current policies of the agents, $R^+$ and $R^-$, when agent $n$ completes task $m$ and when task $m$ was never completed respectively. As in STCLs, TREMOR computes the *shaping reward* $R^\Delta = (R^+ - R^-) \cdot P_\pi^{e,s^+_{t,m}}$. When the shaping reward is positive (coordination locale c is beneficial, lines 29–32), agents participating in coordination locale $c$ will have their transition functions, $P_i$, modified using heuristics to reflect that in each decision epoch $e \in E$, task m can be moved to $Done$ state from $NotDone$, with probability $P_\pi^{e,s^+_{t,m}}$.

For FTCLs, a heuristic similar to that used for STCLs is applied, updating the local transition functions of each agent. First we "extract" the old probability of transitioning given its action and a status of specific task $m$ that will be done by another agent $j$ from $P_i$. Let $s_t \in S_t$ be the starting task state where task $m$ (that some other agent $j$ executes) is not yet completed, i.e., $s_{t,m} = NotDone$ and $s'^+_t, s'^-_t \in S_t$ be two possible ending task states that differ on the status of execution of task $m$, i.e., $s'^+_{t,m} = Done$ and $s'^-_{t,m} = NotDone$. According to the old function $P_i$, agent $i$ "believed" with a certain probability that task $m$ will be completed by agent $j$ in decision epoch $e$. Initially, $P_i(((e, s_{t,m}), s_i), a_i, ((e', s'^+_{t,m}), s'_i)))$ and $P_i(((e, s_{t,m}), s_i), a_i, ((e', s'^-_{t,m}), s'_i)))$ are given as a domain input, and used for updating $P_i$. Now, agent $j$ is shaping the transition function of agent $i$ and it makes agent $i$ "believe" that task $m$ will be completed decision in epoch $e$ with a different probability. Agent $j$'s commitment to the completion of its task $m$ has changed, i.e., task $m$ will now be completed in decision epoch $e$ with probability $P_\pi^{e,s^+_{t,m}}$ – agent $i$'s new transition function $P_i$ should then be updated with this new information. For a given task $m$, we typically have the explicit task status pair, $(s_{t,m}, s'_{t,m})$. We calculate $P_\pi^{e,s^+_{t,m}}$ for each $(s_{t,m}, s'_{t,m})$ separately and keep updating $P_i$ iteratively for all tasks. Formally:

$$P_i \leftarrow P_\pi^{e,s^+_{t,m}} \times P_i(((e, s_{t,m}), s_i), a_i, ((e', s'^+_{t,m}), s'_i)))$$

$$+(1 - P_\pi^{e,s^+_{t,m}}) \times P_i(((e, s_{t,m}), s_i), a_i, ((e', s'^-_{t,m}), s'_i)))$$

We could further generalize this updating step by summing over all current and future task states, however, that would increase the complexity of transition function shaping.

In contrast, if the shaping reward is not beneficial (lines 33–36) agents will have their transition functions $\mathcal{P}_i$ modified to reflect that $s_{t,m}$ cannot change from $NotDone$ to $Done$ in any decision epoch. At last, whenever agent $n$ can execute task $m$, the current shaping reward $R^\Delta$ is added to its local reward function $\mathcal{R}_n$, to either encourage (if $R^\Delta > 0$) or discourage (if $R^\Delta < 0$) agent $n$ from executing task $m$. The algorithm terminates the task assignment evaluation if the current joint policy cannot be improved, i.e., all the sets $I_n$ for $1 \leq n \leq N$ are empty or the number of model refinements is greater than maximum number of iterations.

**Algorithm 1** TREMOR-EvalTaskAssignment(*Agents*, *Tasks*)

1: **for** agent $n = 1, \ldots, N$ **do**
2: $\mathcal{POMDP}_n \leftarrow$ CONSTRUCTPOMDP$(n, Tasks[n])$
3: $\pi_n \leftarrow$ SOLVEPOMDP$(\mathcal{POMDP}_n)$
4:
5: **repeat**
6: **for** agent $n = 1, \ldots, N$ **do**
7: $I_n \leftarrow \emptyset$
8: **for** $c = \langle (s_g, s_{n_1}, ..., s_{n_K})(a_{n_1}, ..., a_{n_K}) \rangle \in CL_s$ such that $n \in \{n_k\}_{1 \leq k \leq K}$ **do**
9: $R^- \leftarrow EU(\pi_n)$
10: $R^+ \leftarrow EU(\pi_n$ when $\mathcal{P}_n$ and $\mathcal{R}_n$ are redefined in terms of joint functions $P$ and $R$ for arguments $((s_g, s_n), a_n, (s'_g, s'_n))$ for all $(s'_g, s'_n) \in S_g \times S_n)$
11: $\hat{c} \leftarrow P_\pi((s_g, s_{n_1}, ..., s_{n_K})(a_{n_1}, ..., a_{n_K}))$
12: $R^\Delta \leftarrow (R^+ - R^-) \cdot \hat{c}$
13: **if** $R^\Delta > 0$ **then**
14: $I_n \leftarrow I_n \cup \{n_k\}_{1 \leq k \leq K}$
15: **for** agent $i \in \{n_k\}_{1 \leq k \leq K}$ and $(s'_g, s'_i) \in S_g \times S_i$ **do**
16: $\mathcal{R}_i((s_g, s_i), a_i, (s'_g, s'_i)) \xleftarrow{+} R^\Delta / K$
17: $\mathcal{P}_i \leftarrow \mathcal{P}_i$ redefined in terms of $P$ for arguments $((s_g, s_i), a_n, (s'_g, s'_i))$ for all $(s'_g, s'_i) \in S_g \times S_i$
18: **else if** $R^\Delta < 0$ **then**
19: $I_n \leftarrow I_n \cup (\{n_k\}_{1 \leq k \leq K} \setminus \{n\})$
20: **for** agent $i \in \{n_k\}_{1 \leq k \leq K} \setminus \{n\}$ and $(s'_g, s'_i) \in S_g \times S_i$ **do**
21: $\mathcal{R}_i((s_g, s_i), a_i, (s'_g, s'_i)) \xleftarrow{+} R^\Delta / (K - 1)$
22: **for** $c = \langle (m, (s_{n_1}, ..., s_{n_K})(a_{n_1}, ..., a_{n_K})) \rangle \in CL_f$ such that $m \in Tasks[n]$ **do**
23: **for all** $e \in E$ **do**
24: $P_\pi^{e,s^+_{t,m}} \leftarrow \sum_{\substack{((e,s_t),s_1,...s_N) \in S \\ : s_{t,m} = Done; a \in A}} P_\pi(s, a)$
25: $R^+ \leftarrow \sum_{k=1}^{K} EU(\pi_{n_k}$ given that task $m$ will be completed in epoch $e \in E$ with probability $P_\pi^{e,s^+_{t,m}})$
26: $R^- \leftarrow \sum_{k=1}^{K} EU(\pi_{n_k}$ if task $m$ not completed)
27: $\hat{c} \leftarrow \sum_{s_g \in S_g} P_\pi((s_g, s_{n_1}, ..., s_{n_K})(a_{n_1}, ..., a_{n_K}))$
28: $R^\Delta \leftarrow (R^+ - R^-) \cdot \hat{c}$
29: **if** $R^\Delta > 0$ **then**
30: $I_n \leftarrow I_n \cup \{n_k\}_{1 \leq k \leq K} \cup \{n\}$
31: **for** agent $i \in \{n_k\}_{1 \leq k \leq K}$ and $e \in E$ **do**
32: Modify $\mathcal{P}_i$ knowing that in each epoch $e \in E$, $s_{t,m}$ can change from $NotDone$ to $Done$ with probability $P_\pi^{e,s^+_{t,m}}$)
33: **else if** $R^\Delta < 0$ **then**
34: $I_n \leftarrow I_n \cup \{n_k\}_{1 \leq k \leq K} \cup \{n\}$
35: **for** agent $i \in \{n_k\}_{1 \leq k \leq K}$ **do**
36: Modify $\mathcal{P}_i$ knowing that in each epoch $e \in E$, $s_{t,m}$ cannot change from $NotDone$ to $Done$
37: **for all** $((s_g, s_n), a_n, (s'_g, s'_n)) \in (S_g \times S_k) \times A_n \times (S_g \times S_k) : s_t$ differs from $s'_t$ on $s_{t,m} \neq s'_{t,m}$ **do**
38: $\mathcal{R}_n((s_t, s_n), a_n, (s'_t, s'_n) \xleftarrow{+} R^\Delta$
39: **for all** $i \in I_n$ **do**
40: $\pi_i \leftarrow$ SOLVEPOMDP$(\mathcal{POMDP}_i)$
41: **until** $\cup_{1 \leq n \leq N} I_n = \emptyset$ or maximum iterations

# 5. EMPIRICAL RESULTS

This section demonstrates that TREMOR can successfully solve DPCL problems orders of magnitude faster than required by existing locally optimal algorithms, while still discovering policies of comparable value. To that end, we evaluate TREMOR's performance on a set of disaster rescue tasks (described in Section 2) by comparing its planning time and solution value with three existing planning approaches.

## 5.1 Experimental Setup

TREMOR employs EVA [18, 19] as the single agent POMDP solver. We compare against JESP (Joint Equilibrium-based Search for Policies) [12] and MBDP (Memory-Bounded Dynamic Programming for DEC-POMDPs) [17], two of the leading approximate algorithms for solving DEC-POMDPs. Lastly, we consider a planner that ignores interactions between agents, i.e. TREMOR without any coordination locales (call independent POMDPs). All planners are given a maximum wall-clock time of 4 hours.

TREMOR and EVA's parameters were set as follows: maximum

iterations of TREMOR= 50, and $\epsilon$ = 5.0. MBDP experiments used the parameters suggested by the authors: type of algorithm = *approximate*, max. number of trees = 3, max. number of observations for the improved MBDP algorithm = 2, depth of recursion = 2, and backup type = *Improved Memory-Bounded Dynamic Programming*. JESP has no tunable parameters.

Experiments were run on quad-core Intel 3.2GHz processors with 8GB of RAM. Each approach was run 20 times on each DPCL and we report the average wall-clock time. For computing expected value of a joint policy, we averaged over 500 runs.

## 5.2 State Space

This set of experiments show that TREMOR can handle large state spaces, unlike existing algorithms. Every experiment has a time horizon of 10, one cleaner robot, and two rescue robots. The state space changes from 81 to 6561 joint states ($2 \times 2$ to $4 \times 10$ grids). Figure 3a shows scaling of TREMOR's runtime with respect to the size of state space. The $x$-axis shows the number of joint states in the problem and the $y$-axis shows $\log$(plan time in sec). MBDP is only able to solve tasks of up to 361 joint states within the time limit and requires 1.5–2.9 orders of magnitude more time than TREMOR. Independent POMDPs plan faster than TREMOR as they disregard all inter-agent interactions.

Figure 3b displays the average reward accrued by polices on the $y$-axis over the same set of tasks as in 3a. TREMOR outperforms MBDP, even though MBDP is an algorithm that plans on the joint models and we expected it to account for interactions better. In addition, TREMOR also achieved the statistically significant result of outperforming independent POMDPs with respect to average reward, although using up to 1.6 orders of magnitude more time ($p < 1.5 \times 10^{-9}$).

TREMOR's runtime does not increase monotonically with the size of the state or horizon as shown in Figure 3. It depends on (i) the time it takes to resolve interactions for each resolution iteration (lines 6–40 in Algorithm 1), (ii) the maximum number of such iterations, both of which change depending on the details of each DPCL.

JESP was unable to solve any task within the time limit and thus is not shown. For illustrative purposes, we ran JESP on a 81 joint state problem with T=2 (reduced from T=10). It finished executing in 228 seconds, yielding a reward of 12.47, while TREMOR required only 1 second and received a reward of 11.13.

## 5.3 Time Horizon

The second set of experiments consider an increasing time horizon from T=2–23, shown in Figures 3c and 3d. These experiments show increased episode lengths lead to higher planning times, but that TREMOR can generate deep joint-policy trees. We considered problems with two rescue robots, one cleaning robot and 361 joint states.

MBDP is able to solve tasks up through T=14, but takes at least 2.6 orders of magnitude more time than TREMOR, while its final policies' rewards are dominated by TREMOR's policies. TREMOR requires at most 1.1 orders of magnitude more time than independent POMDPs, but produces policies that accrue significantly more reward.

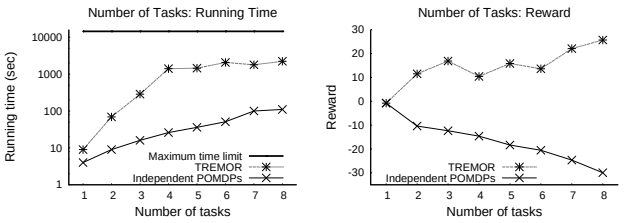


**Figure 4: Agents and Tasks Scale-Up**

## 5.4 Number of Agents and Tasks

The third set of experiments keep the state space and time horizon constant (1089 joint states and T=10) and show that TREMOR scales well with the number of agents. In fact, TREMOR's improvement over independent POMDPs increases with the number of agents. Figure 4a and 4b show the running time and reward accrued on tasks with one cleaning robot and 1–8 rescue robots (the number of victims and rescue robots are equal).

As shown in Figure 4b, TREMOR and the Independent POMDPs' rewards diverge as the number of agents (and tasks) are increased due to increasing numbers of CLs. Increased number of interactions leads to a higher runtime for TREMOR, but also higher rewards. In contrast, the runtime of independent POMDPs do not increase as dramatically, but rewards suffer as they are increasingly penalized for their lack of coordination. MBDP fails to solve any case with two or more tasks within the time limit. [4] TREMOR requires between 0.35 and 1.73 orders of magnitude more time than independent POMDPs, but produces policies that accrue significantly more reward.

## 5.5 Number of CLs

The last set of experiments show how TREMOR performs when the number of CLs changes: more CLs imply more inter-agent interactions, increasing TREMOR's overhead and reducing its benefit relative to MBDP. All experiments have 361 joint states, T=10, two rescue robots, and one cleaning robot; these settings were chosen explicitly so that MBDP could complete the task within the cutoff time. Figure 5a and 5b show the running time and reward with various number of CLs. The performance of TREMOR depends on the number of CLs and maximum number of resolution interactions. As we discussed in the previous section, TREMOR is well-suited for domains which require limited coordination. These results demonstrate that the running time increases and reward decreases when more coordination is required. It should be noted that TREMOR can trade off time and quality by tuning the maximum number of model refinement iterations.

MBDP is able to discover a joint policy superior to TREMOR for very large numbers of CLs. For the problem with the number of CLs = 1368, MBDP received a higher reward than TREMOR and independent POMDPs, although it continues to require more time.

We have shown TREMOR's superior scalability with respect to state space, time horizon, number of agents and tasks, and number of coordination locales. Furthermore, TREMOR provided solutions of comparable, or even superior, quality to those found by existing DEC-POMDP solvers.

[4] We did not try MBDP with one task because there are no interesting same-time coordination locales.

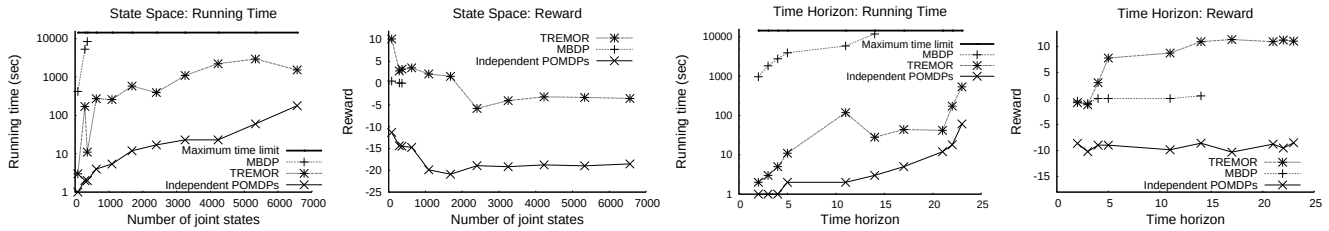


**Figure 3: Comparison with MBDP and Independent POMDPs: State Space and Time Horizon Scale-Up.**

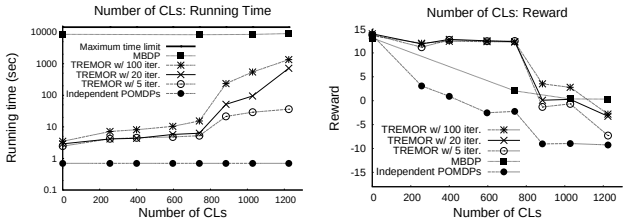


**Figure 5: CLs Scale-Up**

# 6. RELATED WORK AND CONCLUSIONS

As mentioned earlier, others have done significant work to identify classes of DEC-POMDPs that may be solved efficiently. For example, Becker *et al.* [2] assume an individually observable domain where agents are transition independent. ND-POMDPs build on transition-independence and add network structure interactions [9]. Though DPCL assumes individual observability, it differs due to transition dependence (captured using coordination locales), thus focusing on a broad new class of multiagent applications. Task-based ED-DEC-MDPs [1, 4, 10] leverage pre-specified task allocation and dependencies to reduce the search space. This is another key differentiating factor in DPCL, where task allocations and dependencies are not part of the model.

Others have also examined how to combine role allocation with distributed POMDP solvers [13], exploiting problem structure to speed up policy search. Oliehoek *et al.* [14] also exploit problem structure — factored DEC-POMDPs — but assume observation-dependence. TREMOR differs from these and other DEC-POMDP algorithms in its fundamental approach by employing single-agent POMDPs and exploiting social model shaping to manage inter-agent interactions. In this sense, TREMOR shares some similarity with other MDP-related work [7] where subsystems can plan separately, but can iteratively re-plan if the subsystems interact unfavorably. However, the use of POMDPs and social model shaping sets our work apart. Lastly, shaping rewards have been previously used in multi-agent contexts (c.f., Matarić [11]), but are typically present to assist agents via human-specified rewards. In TREMOR, shaping rewards are used to allow coordination between agents without explicit multi-agent planning and are determined autonomously.

This paper has introduced TREMOR, a fundamentally different approach to solve distributed POMDPs. TREMOR is an approximate algorithm and it does not apply to general DEC-POMDPs. However, it is extremely efficient for solving DPCLs, an important subclass of distributed POMDPs. This subclass includes a range of real-world domains where positive or negative agent interactions occur in a relatively small part of the overall state space. By iteratively discovering interactions and using shaping of models to influence efficient individual POMDPs, TREMOR enables a team of agents to act effectively and cohesively in environments with action and observation uncertainty. The main insight behind TREMOR is using social reward and transition shaping allows a DEC-POMDP to be approximated by a set of single-agent POMDPs. TREMOR can thus also exploit advances in single-agent POMDP solvers. Extensive experimental results show how TREMOR provides dramatic speedups over previous distributed POMDP approaches without sacrificing expected reward.

# 7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and suggestions, and Alan Carlin for providing us with the source code for MBDP. This work was supported in part by US. Army SBIR contract number W15P7T-09-C-S601, DARPA SBIR contract number W31P4Q-06-0286, and Perceptronics Solutions, Inc.

# 8. REFERENCES

[1] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov Decision Processes with Event-Driven Interactions. In *AAMAS*, 2004.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *JAIR*, 22, 2004.

[3] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *UAI*, 2000.

[4] A. Beynier and A. Mouaddib. A polynomial algorithm for Decentralized Markov Decision Processes with temporal constraints. In *AAMAS*, 2005.

[5] J. Cooper and M. Goodrich. Towards combining UAV and sensor operator roles in UAV-enabled visual search. In *HRI*, 2008.

[6] D. Goldberg, V. Cicirello, and M. B. Dias. A distributed layerd architecture for mobile robot coordination: Application to space exploration. In *NASA Workshop on Planning and Scheduling for Space*, 2002.

[7] C. Guestrin and G. Gordon. Distributed planning in hierarchical factored MDPs. In *UAI*, 2002.

[8] H. Kitano and S. Tadokoro. RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, March 2001.

[9] J. Marecki, T. Gupta, P. Varakantham, M. Tambe, and M. Yokoo. Not all agents are equal: Scaling up distributed pomdps for agent networks. In *AAMAS*, 2008.

[10] J. Marecki and M. Tambe. On opportunistic techniques for solving decentralized MDPs with temporal constraints. In *AAMAS*, 2007.

[11] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4, 1997.

[12] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[13] R. Nair and M. Tambe. Hybrid BDI-POMDP framework for multiagent teaming. *JAIR*, 23, 2005.

[14] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored DEC-POMDPs. In *AAMAS*, 2008.

[15] M. Petrik and S. Zilberstein. Anytime coordination using separable bilinear programs. In *AAAI*, 2007.

[16] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *AAMAS*, 2005.

[17] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *UAI*, 2007.

[18] P. Varakantham, R. Maheswaran, and M. Tambe. Exploiting belief bounds: Practical POMDPs for personal assistant agents. In *AAMAS*, 2005.

[19] P. Varakantham, R. T. Maheswaran, T. Gupta, and M. Tambe. Towards efficient computation of error bounded solutions in POMDPs: Expected Value Approximation and Dynamic Disjunctive Beliefs. In *IJCAI*, 2007.

[20] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *AAAI*, 2007.

**Table 1: State transition function**

| State | State1 | Action | Probability |
|---|---|---|---|
| Safe | Safe | Success | $P_{safety} \times (1 - P_{actionFailure})$ |
| | | Failure | $P_{safety} \times P_{actionFailure}$ |
| | Unsafe | Success | $(1 - P_{safety}) \times (1 - P_{actionFailure})$ |
| | | Failure | $(1 - P_{safety}) \times P_{actionFailure}$ |
| Unsafe | Safe | Success | $P_{safety} \times (1 - P_{actionFailure})$ |
| | | Failure | $P_{safety} \times P_{actionFailure}$ |
| | Unsafe | Success | $(1 - P_{safety}) \times (1 - P_{actionFailure})$ |
| | | Failure | $(1 - P_{safety}) \times P_{actionFailure}$ |

**Table 2: Reward function**

| Action | Reward |
|---|---|
| Saving the victim *(only rescue robots)* | +8.0 |
| Cleaning debris *(only cleaning robots)* | +1.0 |
| Moving and observing | -0.2 |
| Collisions | -4.0 |
| Dead | -10.0 |

# APPENDIX

# A. DPCL FOR TREMOR

$\langle S, A, P, R, \Omega, O, b \rangle$ with STCLs, FTCLs

(1) S: set of world states (row, column, health): {(0, 0, 0), (0, 1, 0), (0, 2, 0), ..., }

Row and Column: $0 - n$, Health value for each robot: $0 - m$.

(2) $A$: actions: $A$ = {*move north, move east, move south, move west, observe north, observe east, observe south, observe west*}

(3) $P$: state transition function: Transition to a state based on how the health of the robot will be affected due to safety of destination cell ($P_{safety}$) and probability of action failure ($P_{actionFailure}$).

$P_{safety}$: assigned randomly, $P_{actionFailure}$: 0.2 (See Table 1).

In case of collisions between savers *(same time coordination locale, STCL)*, the transition probabilities of states are dependent on actions of other agents. For instance in a collision between two agents in a narrow corridor $(x, y)$, an agent gets to that cell and the other agent goes back to the originating cell. If agents are starting from different cells and colliding in $(x, y)$, this happens with 0.5 probability for each agent.

In case of coordination due to cleaning of debris *(future time coordination locale, FTCL)*, the debris is cleared by the cleaner robot and cleaning action is guaranteed to succeed all the time.

(4) $R$: reward function (See Table 2).

(5) $O$: observations. $O$ = {*Success/Failure* for moving action, *Safe/Unsafe* for observing action} (See Table 3).

(6) STCLs (same-time coordination locales): situations where state or reward resulting from the simultaneous execution of ac-

**Table 3: Observations**

| Action | State | Observation | Probability |
|---|---|---|---|
| *Moving* | Success | Success | 0.8 |
| | | Failure | 0.2 |
| | Failure | Success | 0.6 |
| | | Failure | 0.4 |
| *Observing* | Safe | Safe | 0.8 |
| | | Unsafe | 0.2 |
| | Unsafe | Safe | 0.6 |
| | | Unsafe | 0.4 |

tions.

$cl_s = \langle (s_g, s_{n_1}, \ldots, s_{n_K}), (a_{n_1}, \ldots, a_{n_K}) \rangle$ where $s_g$ is the current global state and $(a_{n_k})_{k=1}^{K}$ are the actions that agents $(n_k)_{k=1}^{K}$ execute in their current local states $(s_{n_k})_{k=1}^{K}$.

(7) FTCLs (future-time coordination locales): situations an action impacts actions in the future.

$cl_f = \langle m, (s_{n_k})_{k=1}^{K}, (a_{n_k})_{k=1}^{K} \rangle$ where $m$ is a task number and $(a_{n_k})_{k=1}^{K}$ are the actions that agents $(n_k)_{k=1}^{K}$ execute in their current local states $(s_{n_k})_{k=1}^{K}$.

## B. EXPERIMENTAL DOMAIN

(1) State Space Scale-Up: 2×2 (# of joint states: 81) – 4×10 (# of joint states: 6561) (See Figure 6).

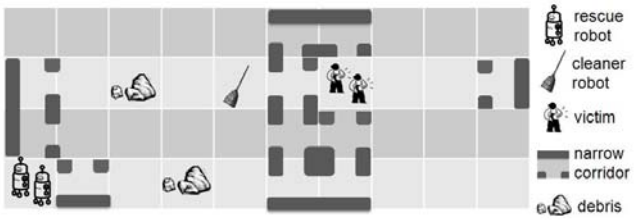


**Figure 6: 4×10 (# of joint states: 6561), T=10: 2 rescue robots, 2 victims, 1 cleaning robot, 2 debris, & 11 narrow corridors.**

One example case both CLs can happen:

$Saver_0$: (3, 0) → (2, 0) → (1, 0) → (1, 1) → (1, 2) → (1, 3) → (1, 4) → (1, 5) → (1, 6)

$Saver_1$: (3, 0) → (3, 1) → (3, 2) → (2, 2) → (2, 3) → (2, 4) → (2, 5) → (1, 5) → (1, 6)

$Cleaner_0$: (1, 4) → (1, 3) → (1, 2)

**STCL happens between savers on (1, 5) at T=7:**

$cl_{s7} : \langle (s_{g7}, s_{14}, s_{25}), (a_1, a_0) \rangle, where$

$s_{g7} : (NotDone_0, NotDone_1, 7)$

$s_{14} : (1, 4, 1)$

$s_{25} : (2, 5, 1)$

$a_1$ : *move east*

$a_0$ : *move north*

**FTCL happens between $Saver_0$ and $Cleaner_0$ at $debris_0$'s location (1, 2):**

$cl_{f1} : \langle m2, (s_{11}, s_{13}), (a_1, a_3) \rangle, where$

$m_2 : cleaningdebris_0at(1, 2)$

$s_{11} : (1, 1, 1)$

$s_{13} : (1, 3, 1)$

$a_1$ : *move east*

$a_3$ : *move west*

(2) Time Horizon Scale-Up: T=2 – 23 (See Figure 7).

(3) Number of Agents and Tasks Scale-Up: 1–8 rescue robots, 1–8 victims (See Figure 8).

(4) Number of Coordination Locales Scale-Up: 0 narrow corridor (# of CLs: 0) – 7 narrow corridors (# of CLs: 1368) (See Figure 9).

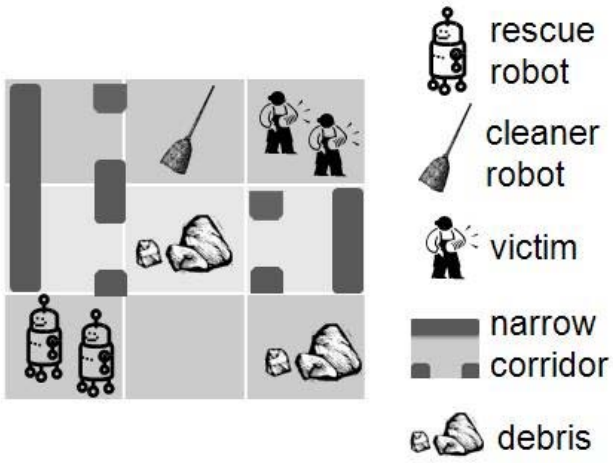


**Figure 7: 3×3 (# of joint states: 361), T=2–23: 2 rescue robots, 2 victims, 1 cleaning robot, 2 debris, & 3 narrow corridors.**

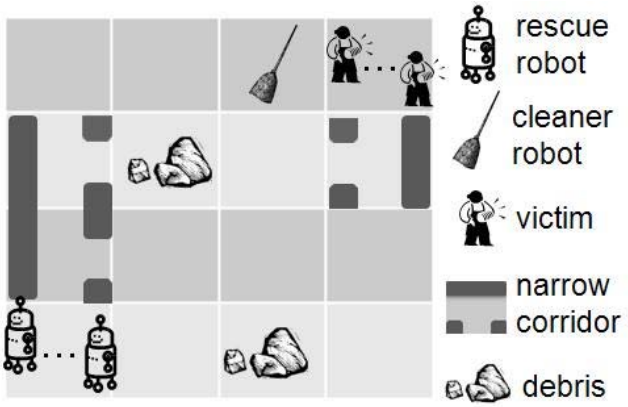


**Figure 8: 4×4 (# of joint states: 1089), T=10: 1–8 rescue robots, 1–8 victims, 1 cleaning robot, 2 debris, & 3 narrow corridors.**

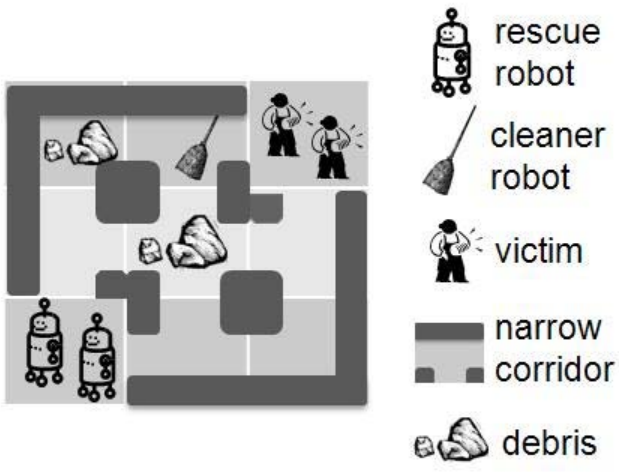


**Figure 9: 3×3 (# of joint states: 361), T=10: 2 rescue robots, 2 victims, 1 cleaning robot, 2 debris, & 0–7 narrow corridors.**

# Introducing Communication in Dis-POMDPs with Finite State Machines

Makoto Tasaki, Yuki Iwanari, Makoto Yokoo, Atsushi Iwasaki, Yuko Sakurai
Kyushu university, Fukuoka 819-0395, Japan
{tasaki, iwanari, sakurai}@agent.is.kyushu-u.ac.jp, {yokoo, iwasaki}@is.kyushu-u.ac.jp

## ABSTRACT

Distributed Partially Observable Markov Decision Problems (Dis-POMDPs) are emerging as a popular approach for modeling sequential decision making in teams operating under uncertainty. To achieve coherent behaviours of agents, performing appropriate run-time communication is essential. Thus, there have been many works on the run-time communication schemes in Dis-POMDPs. Also, a Finite State Machine (FSM) is a popular representation for describing a local policy that works in a very long or an infinite time horizon. In this paper, we examine a run-time communication scheme when the local policy of each agent is represented as an FSM. In this scheme, the meaning of each message is not predefined; it is given implicitly by the interaction between local policies. We propose an iterative-improvement type algorithm that searches for a joint policy where run-time communication incurs some cost. Thus, agents use run-time communication only when doing so is cost-effective. Interestingly, our algorithm can find a joint policy that obtains a better expected reward than a hand-crafted joint policy, and it requires fewer nodes in the local FSM and fewer message types. Furthermore, we experimentally show that our algorithm can obtain a sufficiently large joint policy within a reasonable amount of time.

## 1. INTRODUCTION

Distributed Partially Observable Markov Decision Problems (Dis-POMDPs) are emerging as a popular approach for modeling sequential decision making in teams operating under uncertainty [2, 14, 7].

To achieve coherent behaviours of agents, performing appropriate run-time communication is essential. There have been many works on the run-time communication schemes in Dis-POMDPs. For example, Nair et al. [7] gave the clear semantics of run-time communication when the local policy of an agent is represented as a policy tree. In their scheme, agents communicate their observation/action histories with each other. Thus, they can remove uncertainty about the belief on other agents and they can start from a new synchronized belief state. Shen and Becker [11] proved that synchronizing communication between agents can reduce complexity of Dis-POMDPs. In general, finding an optimal joint policy in Dis-POMDPs is NEXP-Complete [2]. They identified a special case of Dis-POMDPs with communication where the complexity becomes NP-Complete. Goldman and Zilberstein [5] developed a formal model for a decentralized controller, in which an agent has a communication policy as well as an action policy. In this approach, the communication policy of an agent is optimized for a given action policy. Roth *et al*. [10] developed a method for reducing ineffective communications from a joint policy in which run-time communication is performed by default after each observation period.

On the other hand, to describe a local policy that is able to work in very long or infinite time horizons, a Finite State Machine (FSM) is commonly used as a compact representation of a local policy. For POMDPs in an infinite horizon, Poupart and Boutilier [9] developed an algorithm with a non-deterministic FSM, which represents finding an optimal policy as a linear programming formation. [9] introduced a value iteration technique where the expected values of nodes are represented by a vector. This algorithm guarantees the monotonic improvement of the value function while keeping the controller size fixed. FANS [6] algorithm utilizes FSMs to represent policies for Networked Distributed POMDPs in a finite horizon and evaluates policies with dynamic programming and heuristic computation techniques. It can find an optimal policy faster than existing algorithms against increasing number of steps.

Bernstein [1] presented a method for searching a joint policy that corresponds to a correlated equilibrium, i.e., agents can observe a common correlated probabilistic device to achieve better coordination. scheme that utilizes a *correlated device* to achieve better coordination among agents. In [1], an iterative improvement type algorithm is used to modify each local FSM and the correlated device. Szer and Charpillet [13] developed a best-first algorithm to search for an optimal joint policies.

However, as far as the authors aware of, there have been very little work on the run-time communication scheme of Dis-POMDPs when the local policy of each agent is represented as an FSM. Of course, there exists a vast amount of works on communications between FSMs (e.g., [3]), including works on multi-agent systems (e.g., [12]). However, these works intend to use an FSM for formally representing/analyzing a communication protocol, while our work aims to search for a good joint FSM that effectively uses run-time communication.

In this paper, we aim to search for an optimal/semi-optimal joint policy (which is a combination of local FSMs) when the run-time communication incurs some cost. In this scheme, communication is one type of possible actions that can affect the observations of other agents. Agents use run-time communication only when it is cost-effective.

One notable characteristic of this scheme is that the semantics of the messages are not predefined. We just define the possible number of message types without any predefined meanings, e.g., agents can send two types of messages, i.e., either 0 or 1. The meaning of each message type is given implicitly by the interaction between local policies. Since we don't need to define the meaning of communication before hand, this scheme is similar to *cheap talk*,

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

which is considered in game theory literature [4]. However, we assume that the communication incurs some cost, while cheap talk is assumed to have no impact on agents' utilities.

When the number of possible local FSMs (determined by the number of nodes in the FSM, possible actions/observations, and message types) is small, we can exhaustively search for an optimal joint policy. However, this approach quickly becomes infeasible when the number of possible FSMs becomes large. Thus, in this paper, we utilize an iterative-improvement type algorithm, in which, fixing the FSMs of other agents, it tries to improve the FSM of one particular agent and repeats the process for each agent in turn.

Quite interestingly, for two different domains (Tiger-problem and Meeting Problem), our algorithm can find a joint policy that obtains a better expected reward than a hand-crafted joint policy, and it requires fewer nodes in the local FSM and fewer message types. Furthermore, we experimentally show that our algorithm can obtain a sufficiently large joint policy within a reasonable amount of time.

## 2. MODEL: DIS-POMDPS

We follow Bernstein [2] as a description of Dis-POMDPs. Dis-POMDPs with $n$ agents are defined as a tuple: $\langle S, A, P, \Omega, O, R\rangle$. $S$ is a finite set of world states $\{s_1, s_2, \ldots, s_m\}$. $A = \Pi_{1\leq i\leq n} A_i$, where $A_1, \ldots, A_n$ are the sets of actions for agents 1 to $n$. A joint action is represented as $\langle a_1, \ldots, a_n\rangle$. The transition function $P(s_i, \langle a_1, \ldots, a_n\rangle, s_f)$ represents the probability that the current state is $s_i$, if the next state is $s_f$ and the previous joint action is $\langle a_1, \ldots, a_n\rangle$. $\Omega = \Pi_{1\leq i\leq n}\Omega_i$ is the set of joint observations where $\Omega_i$ is the set of observations for agent $i$. The observation function $O(s, \langle a_1, \ldots, a_n\rangle, \omega)$ represents the probability of joint observation $\omega \in \Omega$, if the current state is $s$ and the agents' previous joint action is $\langle a_1, \ldots, a_n\rangle$. We assume that an agent's observations are independent of others' observations. Thus the observation function can be expressed as: $O(s, \langle a_1, \ldots, a_n\rangle, \omega) = O_1(s, \langle a_1, \ldots, a_n\rangle, \omega_1)\cdot\ldots\cdot O_n(s, \langle a_1, \ldots, a_n\rangle, \omega_n)$. Finally, the agents receive a single, immediate joint reward $R(s, \langle a_1, \ldots, a_n\rangle)$ .

Each agent $i$ chooses its actions based on its local policy $\pi_i$. The goal in Dis-POMDPs is to compute a joint policy $\pi = \langle \pi_1, \ldots, \pi_n\rangle$ that maximizes the team's expected reward.

In this paper, we represent a local policy as a Finite State Machine (FSM) defined as follows. While an FSM can be either deterministic or stochastic [1, 9], in this paper, we restrict our attention to deterministic FSMs for simplicity.

A deterministic FSM of agent $i$ is defined as the following tuple: $\langle Q_i, \psi_i, \eta_i, q_i^0\rangle$ where $Q_i$ is the finite set of FSM nodes, $\psi_i : Q_i \rightarrow A_i$ is an action selection function, $\eta_i : Q_i \times O_i \rightarrow Q_i$ is an FSM transition function, and $q_i^0 \in Q_i$ is the starting node of the FSM.

Given FSMs for each agent, we can obtain a joint policy, i.e., one FSM that combines all local FSMs. The expected discount reward of this joint FSM can be obtained by solving a system of linear equations defined over $V_j(s)$ for each $s \in S$, where $j$ is the index of a node in the joint FSM.

$$V_j(s) = R(s, \vec{a}) + \gamma \sum_{s'\in S, \omega\in\Omega} P'(s', \omega|s, \vec{a}) V_{j'}(s')$$

where $\gamma$ is a discount factor, $\vec{a} = \langle \psi_0(j), \ldots, \psi_n(j)\rangle$ is a set of actions associated with node $j$, and $P'(s', \omega|s, \vec{a}) = P(s, \vec{a}, s')$ $\sum_{\omega\in\Omega \ s.t. \ \eta(j,\omega)=j'} O(s, \vec{a}, \omega)$.

**Table 1: Transition function $P$: * corresponds with either *OpenLeft* or *OpenRight***

| Action/Transition | $SL \rightarrow SL$ $(SR \rightarrow SR)$ | $SL \rightarrow SR$ $(SR \rightarrow SL)$ |
|---|---|---|
| $\langle *, * \ or \ Listen\rangle$ | 0.5 | 0.5 |
| $\langle * \ or \ Listen, *\rangle$ | 0.5 | 0.5 |
| $\langle Listen, Listen\rangle$ | 1.0 | 0.0 |

**Table 2: Observation function $O$: * of state corresponds with either *SL* or *SR*, * of action corresponds with *OpenLeft* or *OpenRight***

| State | Action | $HL$ | $HR$ | $Reset$ |
|---|---|---|---|---|
| $SL$ | $\langle Listen, Listen\rangle$ | 0.85 | 0.15 | 0.0 |
| $SR$ | $\langle Listen, Listen\rangle$ | 0.15 | 0.85 | 0.0 |
| $*$ | $\langle *, * \ or \ Listen\rangle$ | 0.0 | 0.0 | 1.0 |
| $*$ | $\langle * \ or \ Listen, *\rangle$ | 0.0 | 0.0 | 1.0 |

**Table 3: Reward function $R$: $c$ corresponds with listen cost, $d$ corresponds with tiger cost, and $d/2$ denotes the cost when agents jointly open door to tiger.**

| Action/State | $SL$ | $SR$ |
|---|---|---|
| $\langle OpenRight, OpenRight\rangle$ | $+20$ | $-d/2$ |
| $\langle OpenLeft, OpenLeft\rangle$ | $-d/2$ | $+20$ |
| $\langle OpenRight, OpenLeft\rangle$ | $-d$ | $-d$ |
| $\langle OpenLeft, OpenRight\rangle$ | $-d$ | $-d$ |
| $\langle Listen, Listen\rangle$ | $-c$ | $-c$ |
| $\langle Listen, OpenRight\rangle$ | $+10$ | $-d$ |
| $\langle OpenRight, Listen\rangle$ | $+10$ | $-d$ |
| $\langle Listen, OpenLeft\rangle$ | $-d$ | $+10$ |
| $\langle OpenLeft, Listen\rangle$ | $-d$ | $+10$ |

## 3. ILLUSTRATIVE DOMAIN: MULTI-AGENT TIGER PROBLEM

We introduce an illustrative domain called the multiagent tiger problem [7], in which performing appropriate run-time communication is effective. Two agents stand in front of the two rooms and its doors labeled "left" and "right". Behind one door lies a hungry tiger and behind the other lies untold riches but the agents do not know the position of either. Thus, $S = \{SL, SR\}$, indicating behind which door the tiger is present. The agents can jointly or individually open either door. In addition, they can independently listen for the presence of the tiger. Thus, $A_0 = A_1 = \{$*OpenLeft, OpenRight, Listen*$\}$. The transition function $P$ specifies that every time either agent opens one of the doors, the state is reset to $SL$ or $SR$ with equal probability, regardless of the action of the other agent (see Table 1). However, if both agents choose *Listen*, the state is unchanged.

The observation function $O$ will return either $HL$ or $HR$ with different probabilities depending on the joint action taken and the resulting world state (see Table 2). For example, if both agents listen and the tiger is behind the left door (state is $SL$), each agent receives observation $HL$ with probability $0.85$ and $HR$ with probability $0.15$. Reward function $R$ returns a joint reward (see Table 3). For example, the injury sustained if they opened the door to the tiger is less severe if they open that door together than if they open the door alone.

## 4. RUN-TIME COMMUNICATION SCHEME AND SEARCH ALGORITHM

In this paper, we aim to search for an optimal/semi-optimal joint policy when the run-time communication incurs some cost. In this scheme, communication is one type of possible actions that can affect the observation of other agents. Agents use run-time communication only when doing so is cost-effective. More specifically, we simply define the possible number of message types without any predefined meanings, because the meaning of each message type is given implicitly by the interaction between local policies.

## 4.1 Run-time communication scheme

We illustrate how run-time communication among FSMs works using the multiagent tiger problem. Let us assume agents can communicate two types of messages, 0 and 1, which correspond to actions *Comm0* and *Comm1*, respectively.

We assume the communication action of one agent can supersede the other agent's *Listen* action, but not *OpenLeft* and *OpenRight*. More specifically, when agent 0 communicates 0 (or 1), and agent 1 listens or communicates, then agent 1 observes *Signal0* (or *Signal1*). On the other hand, if agent 1 does *OpenLeft* (or *OpenRight*), the state is reset and both agents observe *Reset*. When agent 0 communicates and agent 1 listens, then agent 0 receives no observation (denoted as *Nothing*).

We assume communication actions require a certain cost. Also, by choosing a communication action, an agent might loose an opportunity to perform other more profitable actions.

Note that we don't ascribe any predefined meanings to the messages, such as message 0 means that the agent observes that the tiger is left, etc. However, the search algorithm described in the following subsection can automatically find a joint policy in which these messages are used effectively. For example, agent 0 communicates message 0 (or message 1) if he observes the tiger is left (or right), and agent 1 chooses an appropriate action based on the received message type.

## 4.2 Search algorithm

The number of possible local FSMs is determined by the number of nodes in the FSM, the number of possible actions/observations, and the number of message types. When the number of possible local FSMs is small, we can exhaustively search for an optimal joint policy, or use more sophisticated search algorithms [13]. However, finding an optimal joint policy becomes quickly infeasible when the number of possible FSMs becomes large. Thus, in this paper, we introduce an iterative-improvement type algorithm that can find a locally optimal joint policy quickly.

There exist several works on iterative improvement type search algorithms for FSMs, such as [1]. Since we assume deterministic FSMs, we cannot directly apply the techniques introduced in [1]. Therefore we use a relatively simple iterative improvement type algorithm similar to JESP [8]. It changes one node of one particular agent one by one, by using hill-climbing procedure.

Algorithm 1 describes our algorithm. First, it generates an initial default FSM for each agent (Line 1). $\Pi$ contains the current joint FSM. Then, the algorithm picks one agent $i$ in turn (Line 5) and chooses one particular node $q$ of the FSM of agent $i$ (Line 6). Then, the algorithm tries all possible local FSMs by changing only this particular node $q$ and chooses the one that gives the highest expected reward (Lines 7 to 13). This process is repeated for each agent in turn, until no improvement is obtained.

The quality of the obtained local optimal joint policy critically depends on the choice of the initial default FSMs. Thus, we repeatedly run Algorithm 1 starting from different initial default FSMs and choose the best joint policy obtained in the multiple trials.

**Algorithm 1** NODE-BASE-APPROXIMATION($Q$)

```
1: Π ← GETDEFAULTFSM(Q)
2: V_tmp ← V(Π)
3: V_max ← V_tmp
4: Π_best ← Π
5: for all agent i do
6:   for all q ∈ Q_i do
7:     for all a ∈ A_i do
8:       Expand ψ_i : q → a
9:       for all η_i' ∈getAllFSMTransition(q, Q_i, Ω_i) do
10:        η ← η'
11:        if V(Π) − V_tmp > 0 then
12:          V_tmp ← V(Π)
13:          Π_best ← Π
14: if V_tmp − V_max > 0 then
15:   V_max ← V_tmp
16:   Go to line 5
17: return Π_best
```

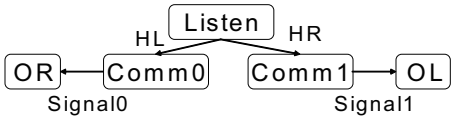


**Figure 1: The hand-crafted policy represented by FSM in the Tiger Problem.**

# 5. EXPERIMENTAL RESULTS

In this section, we empirically evaluate our proposed algorithm in Section 4.1 using the tiger and meeting problems in terms of the expected reward and runtime of the obtained policies. These problems are slightly extended to treat communication among agents, since our proposed algorithm introduces communication among agents. We ran the algorithm for an initial state chosen randomly, and the initial/default policy is also selected randomly for all states. Then, we perform an empirical comparison of the obtained policy and a hand-crafted policy on those two problems. The discount factor $\gamma$ is set to 0.9.

## 5.1 Tiger problem

Let us describe a hand-crafted policy for the tiger problem. Initially, we thought we needed (at least) two types of messages to make communication worthwhile in the tiger problem, i.e., message 0/1 means the agent observed *HL*/*HR*. Figure 1 shows a hand-crafted policy based on this idea. The initial node is the one with a bold line and, each edge illustrates a transition according to an observation. Though we omit a transition back to the initial node, when an agent observes *Reset*, the state always transits to the initial node. In the policy, both agents first chooses *Listen*, and they send either message 0 or 1 according to its observation (*HR* or *HL*).

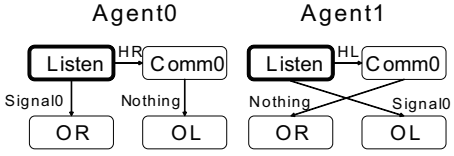


**Figure 2: The obtained FSM with one type of communication**

**Table 4: The comparison of results in the global and local search algorithms**

| | Runtime [secs] | Expected reward |
|---|---|---|
| Global search | 8897.0 | 17.19 |
| Local search | 690.00 | 17.19 |

**Table 5: Evaluating the expected rewards and runtime by increasing the number of FSM nodes**

| Number of nodes | 4 | 5 | 6 | 7 |
|---|---|---|---|---|
| Expected reward | 9.35 | 20.14 | 26.77 | 26.77 |
| Average runtime[sec] | 0.14 | 1.70 | 7.24 | 22.98 |

After sending messages, if they realize they had the observation of *HR* (*HL*), they choose *OpenLeft* (*OpenRight*). If they have different observations, both move back to the initial node. This policy is intuitively quite natural and it seems difficult to generate any reasonable policy using fewer nodes/message types than the hand-crafted policy.

However, our algorithm finds a better policy with only one message type and four nodes. Figure 2 shows the obtained FSM by our algorithm. When we ran the algorithms, we assume that the tiger cost is $d = 50$, the listen cost is $c = 2$, and the communication cost is $e = 2$. In the setting, Figure 1 achieves the average expected reward of $14.58$, and Figure 2 achieves that of $17.19$. Let us explain why the obtained FSM outperforms the hand-crafted FSM in terms of the expected rewards. In the obtained FSM, agent 0 communicates only when he observes *HR*. Also, agent 1 communicates only when he observes *HL*. When agent 0 observes *HL*, he chooses *Listen* again. If agent 1 communicate in the next step, it means that agent 1 observes *HL* (since agent 1 communicates only when he observes *HL*), thus both agents choose *OR*. Thus, this policy can achieve the same coherent behavior as the hand-crafted policy using only one message type and four nodes. Furthermore, the obtained FSM is more efficient than the hand-crafted policy when the observations of agents contradict. When agent 0 observes *HR* and agent 1 observes *HL*, the obtained FSM chooses the same action as the hand-crafted policy.

On the other hand, when agent 0 observes *HL* and agent 1 observes *HR*, the obtained policy immediately returns to the initial node without any communication. Thus, the obtained FSM outperforms the hand-crafted FSM since the hand-crafted FSM requires one extra step in this case. We perform an empirical comparison of the global and local search algorithms using the tiger problem. Each agent has an FSM with four nodes and uses only one type of communication (*Comm0*).

Table 4 shows the comparison of results in the global and local search algorithms. In the local search algorithm, we perform 5000 time experiments and choose a solution (joint policy) with the highest expected reward among 5000 obtained policies. The expected reward of the joint policy in Figure 2 is $17.19$. The joint policy is equivalent to the one obtained by utilizing the global search algorithm. The total runtime for the global search algorithm to find the joint policy is $8897$ seconds, while that for the local search algorithm is $690$ seconds. The runtime for the local search algorithm per one experiment is $138$ msec. When a policy has a few nodes in an FSM, the local search algorithm can find a joint policy whose expected reward is equivalent to that of the obtained joint policy from the global search algorithm, though the runtime is reduced.

Finally, to examine the scalability of our algorithm, we evaluate

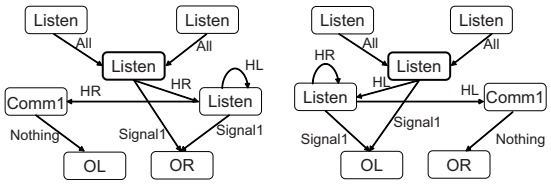


**Figure 3: The obtained FSM with seven nodes**

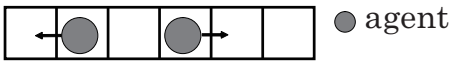


**Figure 4: Meeting problem (2agents, $l = 6$)**

the expected rewards and runtime by increasing the number of FSM nodes from 4 to 7. Each agent uses two types of communication for $d = 1000$ and $c = 0.1$. We perform 1000 time experiments for each number of nodes and choose the joint policies with the highest expected reward among the 1000 obtained policies shown in Table 5. The expected rewards are $9.35$ with $4$ nodes and $26.77$ with $6$ and $7$ nodes. The FSM with $4$ nodes generated the policy equivalent to Figure 2. The FSM with 7 nodes generated the policy shown in Figure 3. However, since two of the seven nodes are never reached when the FSM is conducted, only five nodes in the FSM with 7 nodes are used to control agents.

Table 5 denotes that the expected reward of the FSM with five nodes ($20.14$) is outperformed by that of the one with seven nodes ($26.77$). In particular, though the FSM with seven nodes includes two unreachable nodes, it achieves better expected rewards. Since the average runtime for seven nodes is $22.98$ seconds, our algorithm can generate a joint policy in a feasible time when the number of nodes is increased. Thus, we can say that our algorithm has adequate scalability.

## 5.2 Meeting problem

This subsection focuses on a meeting problem where two agents try to meet on a line field of length $l$ with no obstacles [13]. Here, each agent has the available actions of move left, move right, or communicate on the current square. An agent cannot realize where his partner is and he can observe an end wall of the line field only when he reaches it. Their goal is to stay on either the same or their neighbor's square. When their goal is achieved, they receive a reward of 1.0.

There are six observations. First, if an agent moves left or right, he observes a wall to his left (*LeftEdge*), his right (*RightEdge*), or no wall (*Middle*). Second, an agent observes *Signal* if his partner agent chooses *Comm*. Third, if one agent chooses *Comm* and the other chooses a different action from *Comm*, the agent which chose *Comm* observes *Nothing*, and the other agent observes *Signal*. In other words, if agent $0$ chooses to communicate, the message is delivered to agent $1$ regardless of the choice of actions of agent $1$. Finally, if both agents meet, they receive the reward and observe *Reset*. The positions of the agents are shuffled randomly and the next period begins.

We perform an empirical comparison of the obtained policy and the hand-crafted policies on the meeting problem. Let us describe the two types of hand-crafted FSMs with/without communication. Figure $5$ illustrates an FSM without communication, where one agent moves left and the other agent moves right until they reaches either edge. Then, the agent who reaches an edge changes his di-

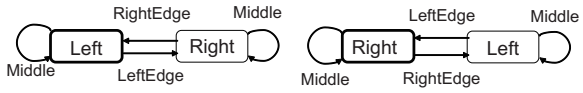


**Figure 5: The hand-crafted FSM without communication**

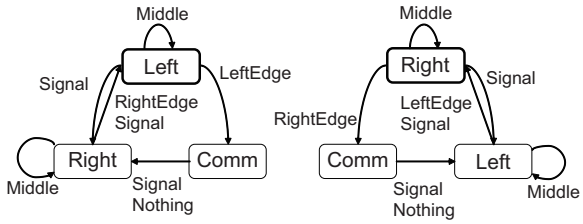


**Figure 6: The hand-crafted FSM with communication**

rection. Next, we describe a policy to make communication worthwhile in the meeting problem described in Figure 6. The policy is basically the same as the policy without communication until one agent reaches the edge. If agent 0, who is moving left, reaches the edge before meeting agent 1, it means that agent 1 is on his right side and keeps moving to right. Thus, agent 0 communicates (send a signal to) agent 1 the fact that agent 0 is at the left edge. By receiving this signal, agent 1 turns and starts moving to the left.

Since these two FSMs are natural and reasonable, it seems difficult to outperform these hand-crafted FSMs. However, our algorithm finds a policy (shown in Figure 7) that outperforms them when $l$ is small. In this policy, both agents choose to communicate in every two steps until one agent reaches the edge. At first glance, this sounds wasteful, since by communicating, agents lose one step to move. However, in many configurations, the agents are moving in opposite directions. Therefore, moving cautiously can pay off. If agent 0 reaches an edge, he changes his direction without communication and he will not communicate afterward. Agent 1, who observed that agent 0 did not communicate, changes his direction and will not communicate afterward. In the meeting problem with $l = 6$, the expected reward of Figure 6 is $3.61$, while Figure 5 is $3.92$ and Figure 7 is $4.15$. More specifically, Figure 8 illustrates how these FSMs work. The left side of Figure 8 shows the hand-crafted FSM without communication (*no-comm*), the middle shows the hand-crafted FSM with communication (*comm-hand*), and the right side shows the obtained FSM (*comm-search*). At the initial state, *no-comm* needs 5 steps to meet the agents. Also, the left agent of *comm-hand* communicates at step 2, since he reaches the left edge, while staying in the same position. Furthermore, the right agent moves to the right square.

On the other hand, if an initial state is drawn randomly, the expected reward of *comm-search* outperforms that of *comm-hand*. Agents with *comm-search* meet in four steps. At the first step, one agent moves left and the other moves right. Since the agent who chose move left faces an edge, at the second step, he chooses move right, while the other agent communicates. Then, the right side agent receives no signal from the other agent. At the third step, he changes his direction to move, i.e., moves left. Since the left side agent receives a signal, he keeps to the right. As a result, the agents meet at the fourth step. Considering the possible initial states in $l = 6$, it is likely that *comm-search* requires fewer steps to meet than *comm-hand*.

Figure 9 describes the expected rewards of the obtained and the

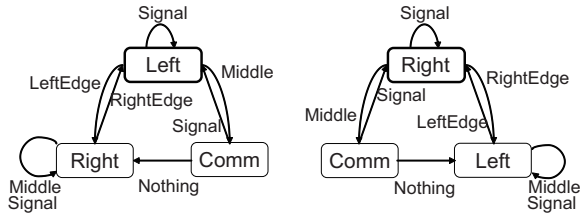


**Figure 7: The obtained FSM in meeting problem**

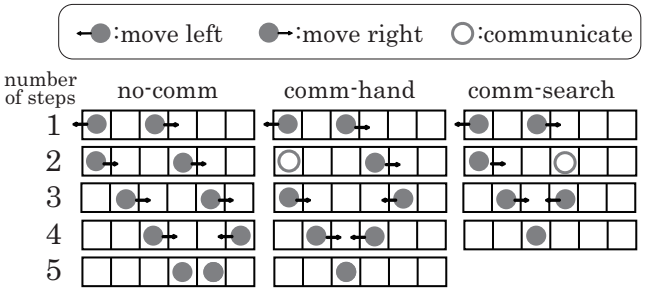


**Figure 8: The comparison of the obtained FSM and the hand-crafted FSMs $l = 6$**

two hand-crafted FSMs (*comm-search*, *comm-hand*, and *no-comm*), varying in the lengths ($l = \{5, 6, 7, 8, 9\}$). Here, though *comm-search* outperforms the other FSMs in terms of the expected rewards in $l = \{5, 6, 7, 8\}$, the difference of performances seems small as the length increases. In $l = 9$, the expected rewards of *comm-search* and *comm-hand* are $2.58$. Since both has the same FSM in *comm-hand*. Also in $l \geq 10$, our algorithm finds similar FSM to that of $l = 9$. We choose $l \geq 5$ because when $l \leq 4$, a trivial FSM with one node can maximize the expected reward (e.g., both agents just keep on moving to the left). Therefore, we pay our attention to cases with $l \geq 5$ where it is effective for agents to utilize some nodes in addition to the initial node.

## 6. CONCLUSION

This paper proposed a run-time communication scheme where the local policy of each agent is represented as an FSM. Also, we proposed an iterative-improvement type algorithm that searches for a joint policy where run-time communication incurs some cost. Thus, agents use run-time communication only when doing so is cost-effective.

Interestingly, our algorithm could find a joint policy that obtains a better expected reward than a hand-crafted joint policy that reasonably utilizes a communication in response with an observation. In addition, the obtained policy requires fewer nodes in the local FSM and fewer message types than the hand-crafted joint policy. Furthermore, we experimentally confirmed that our algorithm generates a sufficiently large joint policy within a reasonable amount of time. Future works will examine our algorithm with more agents, or generate global optimal FSMs in larger-scale problems.

## 7. REFERENCES

[1] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized pomdps. In *IJCAI*, pages 1287–1292, 2005.

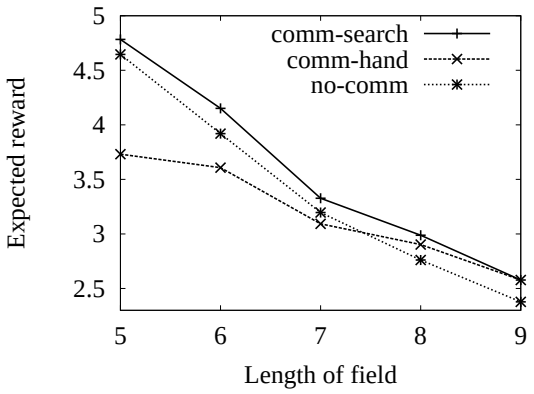


**Figure 9: Evaluating the expected reward for obtained FSM and hand-crafted FSM in meeting problem**


[2] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 32–37, 2000.

[3] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[4] J. S. Crawford, VP. Strategic information transmission. In *Econometrica*, volume 50, pages 1431–1451, 1982.

[5] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 137–144, 2003.

[6] J. Marecki, T. Gupta, P. Varakantham, M. Tambe, and M. Yokoo. Not all agents are equal: scaling up distributed pomdps for agent networks. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 485–492, 2008.

[7] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for improving policy computation in distributed POMDPs. In *AAMAS-04*, pages 1098–1105, 2004.

[8] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Prodeedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 705–711, 2003.

[9] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16 (NIPS-2003)*, 2003.

[10] M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 786–793, 2005.

[11] J. Shen, R. Becker, and V. Lesser. Agent interaction in distributed pomdps and its implications on complexity. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 529–536, New York, NY, USA, 2006. ACM.

[12] M. P. Singh. Formalizing communication protocols for multiagent systems. *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[13] D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon dec-pomdps. In *Proceedings of the 16th European Conference on Machine Learning*, pages 389–399, 2005.

[14] D. Szer and S. Z. Francois Charpillet. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 576–590, 2005.

# Flexible Approximation of Structured Interactions in Decentralized Markov Decision Processes

Stefan J. Witwicki and Edmund H. Durfee
Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109
{witwicki,durfee}@umich.edu

## ABSTRACT

Policy optimization in Decentralized MDPs is in general intractable, so researchers have developed a variety of techniques geared towards solving restricted subclasses of these problems more efficiently. In particular, Becker and colleagues have identified an interesting class wherein agents' interactions consist of event-driven dependencies, and have applied informed policy search techniques to solve these problems optimally. Here we present a dual formulation of the Event-driven DEC-MDP, representing interactions with a well-established commitment paradigm. We claim that, for this particular class of problems, searching a space of rich commitments is equivalent to searching the policy space directly. And we argue that our commitment-based reformulation not only enables more efficient, scalable computation of approximate solutions, but further provides a natural flexibility of approximation by which interactions can be modeled with more or less detail.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

## General Terms

Algorithms, Performance

## Keywords

Multiagent Systems, Decentralized Markov Decision Processes, Event-Driven Interactions, Commitments

## 1. INTRODUCTION

For agents involved in cooperative planning and decision-making, there is often the danger that uncertainty in the timing of one's own actions could carry over to uncertainty in the timing of interactions with others. Our work is motivated by problems in which agents can affect the transitions and rewards of one another, and so benefit from coordinating their actions, but in doing so must account for durational uncertainty in the effects of these actions. These problems have inspired a variety of techniques for successfully solving them in restricted cases [6, 12, 13]. One common theme explored in prior works is the use of temporal decoupling: if there is a range of possible times that an inter-agent effect can take place, temporal decoupling entails selecting individual time points and constraining the effect to occur at (or by) these times.

In this paper, we generalize the notion of temporal decoupling into a framework of *temporal commitments* that can be used to constrain agent interaction modeled by Decentralized Markov Decision Processes (DEC-MDPs). We focus on a class of loosely-coupled DEC-MDPs called Event-Driven DEC-MDPs [2] that represent agent interactions in a structured form of event dependencies. As Becker and colleagues [2] note, even this restricted class is difficult to solve optimally. But here we show that our decoupling methodology can be used to coordinate structured, temporally-uncertain interactions effectively and efficiently, scaling well with the degree of temporal uncertainty.

We begin by providing a detailed description of the Event-Driven problem class along with an example to illustrate how temporally-uncertain interactions are modeled. Next, in Section 3, we describe Becker's optimal solution algorithm and give an overview of related DEC-MDP approaches. In Section 4, we characterize structural properties that can be taken advantage of in solving problems with temporally-uncertain interactions, and discuss their (limited) impacts on Becker's optimal solution algorithm, exposing the need for algorithms that can exploit temporal uncertainty. In Section 5, we describe that commitment semantics that form the core of our methodology and show how they can be extended so as to provide a reformulation of the Event-Driven DEC-MDP problem. In Section 6, we prove (given common assumptions) that our methodology yields optimal solutions equivalent to those returned by Becker's algorithm. And in Section 7, we argue that our methodology can flexibly trade off solution quality and computational complexity, producing approximate solutions more efficiently than Becker's algorithm. We conclude with a preliminary evaluation and discussion.

## 2. PROBLEM DESCRIPTION

Our approach addresses coordination problems with temporal uncertainty and temporal constraints that arise among loosely-coupled agents that interact through features in their shared environment. The kinds of problems where this and the assumptions described below hold arise in the DARPA Coordinators [20] application domain where one agent might establish the preconditions for another agent's action, or where successful achievement of objectives requires the simultaneous execution of actions by multiple agents. These

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

types of multiagent coordination problems can be represented in the TAEMS description language [8]. Here we describe one such problem (in Section 2.1) and model the problem as a DEC-MDP (in Section 2.2).

## 2.1 Example

An example of a TAEMS problem is depicted in Figure 1. Consider two autonomous vehicle agents with various mission objectives, one of which is to find and retrieve resources in their environment. The two agents have differing abilities, so in order to fulfil their objectives, they must work together to first locate resource $X$ and then collect resource $X$. The problem is made more difficult by uncertainty in the timing details of the mission. Depending on where $X$ is located, the UAV agent may have to fly around for a variable amount of time before finding it. Similarly, depending on what obstacles lie in its path, the UGV agent may take a variable amount of time to obtain the item after it has been located by the UAV agent.

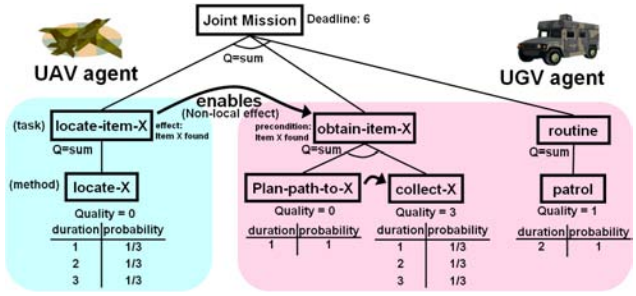


**Figure 1: Autonomous Vehicle Example Problem**

More formally, the joint mission objectives are represented by a hierarchy of *tasks* the agents can perform. The agents accumulate *quality* by completing tasks, the quality of each of which is dictated by a function (in this case, summation) of the qualities of the underlying completed tasks. A task may have *preconditions* necessary for starting the task and ensuring its successful completion. A task may also have *effects* that are realized upon its successful completion. Each task also has a probability distribution over *duration*, modeling temporal uncertainty. And each task has a *deadline*, which is the last time step at which the task can yield its prescribed quality and effects. If the task completes later than its deadline, then the agents do not benefit from it.

Agent interaction occurs with the fulfillment of task dependencies. The effects of one agent's task may meet preconditions for a task of another agent, signifying that a *non-local effect* (*NLE*) exists between the two tasks. The second agent is non-locally affected by the first agent completing its task. We call this kind of non-local effect an *enablement*: *locate-item-X* enables *obtain-item-X*. So we are faced with the problem of coordinating the behavior of the agents so that they perform tasks that will achieve mission objectives and maximize their collective rewards within deadlines.

## 2.2 Event-Driven DEC-MDPs

This problem can be modeled using a DEC-MDP with Event-Driven Interactions [2] (or Event-Driven DEC-MDP, for short). As defined by Bernstein and colleagues [4], an $n$-agent **DEC-MDP** can be described by the tuple $\langle S, A, P, R, \Omega, O \rangle$, where $S$ is a set of world states, $A = A_1 \times A_2 \times ... \times A_n$ is the joint action space, $P : S \times A \times S \rightarrow [0, 1]$ is the transition function, $R : S \times A \times S \rightarrow \mathbb{R}^n$ is the joint reward function, $\Omega = \Omega_1 \times \Omega_2 \times ... \times \Omega_n$ is the joint observation space, and $O : S \times A \times S \times \Omega \rightarrow \mathbb{R}$ is the observation function.

In our example problem from Figure 1, the actions available to each agent are to begin executing each of its methods or to *wait* a time step. The transition probabilities associated with each of these method-execution actions are dictated the respective durational outcomes. The reward for entering a state in which a task has just been completed (by its deadline and with its dependencies previously fulfilled) is the prescribed quality of that task (as given in Figure 1). The agents observe whether or not their own tasks have completed as well as those tasks which affect the executions of their own tasks.

Event-driven DEC-MDP world state is factored into *local state* components so as to separate features relevant to one agent from features relevant to other agents. In our TAEMS problem, an agent's local state is composed of the current time as well as features relating to the executions and outcomes of its methods. For simplicity, we consider features dictating whether or not an agent's task is enabled to be part of that agent's local state, and therefore fully observable by that agent. Although this is not strictly speaking how Becker models TAEMS problems in his Event-Driven DEC-MDP treatment [2], he notes that this added observability could be incorporated into his model.

In addition to being *factored*, *locally fully observable*, and *reward independent*, Event-Driven DEC-MDPs have structured transition dependencies [2]. In particular, one agent may influence the local state transitions of another through the occurrence of a *proper event*, a disjunction of primitive events of the form $e = (s_i, a_i, s_i')$, each of which may occur in agent $i$'s execution history. Interactions occur though event *dependencies* of the form $d_{ij}^k = \langle E_i^k, D_j^k \rangle$, whereby an event in $E_i^k$ brings about a change in the transitions, $D_j^k$ (which is made up of state-action pairs), of agent $j$. Dependency satisfaction is captured by Boolean variable $b_{s_j a_j}^k$, which is true when an event in $E_i^k$ has occurred.

The transition function $P$ of a DEC-MDP with event-driven interactions is defined as the combination of individual transition functions $P_i\left(s_i' | s_i, a_i, b_{s_i a_i}^k\right)$, one for each agent. That is, the agents' *local state* transitions are independent of one another with the exception of event dependencies captured by the $b_{s_i a_i}^k$ variables.

In our example problem from Figure 1, we can model the *enablement* NLE between *locate-item-X* and *obtain-item-X* by a set of event dependencies $\left\{d_{12}^k\right\}$. For example, $d_{12}^2 = \left\langle E_1^2, D_2^2 \right\rangle$ represents the dependency satisfied by *locate-item-X* completing at or before time 2. $D_2^2$ thus contains state-action pairs representing agent 2 attempting to execute *obtain-item-X* at time 2. There is one such dependency for each time step that agent 2 could be enabled.

# 3. EXISTING SOLUTION METHODS

## 3.1 Dependency-Augmented MDPs and CSA

Becker's algorithm [2] exploits dependency structure in the Event-Driven DEC-MDP model. Becker shows that, because interactions are restricted to event dependencies, the DEC-MDP can be broken into augmented local MDPs, each representing additional information regarding the agent's

(non-local) dependencies. $S_i$ is expanded to contain variables that represent the dependency history in all states. Becker's algorithm constructs each augmented local MDP $\langle S'_i, A_i, P_i, R_i \rangle_{\pi_{-i}}$ by assuming that the policies of the other agents ($\pi_{-i}$) have been fixed, extracting dependency information (in the form of $P\left(d_{ij}^k | s_i\right)$) from these policies, and then incorporating this dependency information into the local state and local transitions. Further, the reward function of the augmented local MDP is supplemented with the expected nonlocal reward obtained by other dependent agents resulting from satisfaction of local-event dependencies.

The DEC-MDP is thereby converted into a set of local MDPs, each with transitions and rewards dependent on a parametrization of the policies of the other agents and rewards that reflect the joint utility of the system (as a linear combination of those parameters). This allows for the application of Becker's coverage set algorithm (CSA) [3]. The coverage set algorithm has been shown in the past to find optimal solutions tractably by performing an informed search of the joint policy space, but there is additional computational cost in applying it to Event-Driven DEC-MDPs, as we will discuss in Section 4. There has been recent development of a more efficient reformulation of CSA using Separable Bilinear Programming [15] but no publications, as of yet, extend it to handling Event-Driven interactions.

## 3.2 Other Related Work

The general class of DEC-MDPs has been shown by Bernstein and colleagues to be NEXP-complete [4]. Despite this daunting complexity result, researchers have developed optimal algorithms including A* Heuristic search [18], dynamic programming [11], and policy iteration [5], as well as a variety of approximate algorithms [1, 5, 16]. All of these methods are applicable to DEC-MDPs with Event-Driven interactions, but as they do not exploit the special dependency structure, they are computationally intractable for all but the smallest of problems.

The methodology that we present here, like Becker's, gains traction by taking advantage of structure in problems with Event-Driven interactions. Along these lines, researchers have developed practical techniques that exploit structure in other DEC-MDP problem subclasses. Goldman and Zilberstein have developed a suite of techniques that exploit goal-oriented behavior in decentralized MDPs (and POMDPs) under different communication regimes [10]. Nair extended his JESP algorithm to solve a particular class of transition-independent DEC-POMDPs called network distributed POMDPs (ND-POMDPs), taking advantage of the transition independence in the best-response calculations as well as locality of interactions within the system of agents [14].

Our work also has the flavor of decoupling the agents' problems by imposing temporal constraints on their interactions. Others have recently taken similar temporal decoupling approaches. Beynier [6] and, more recently, Marecki [13] have also looked at another special class of DEC-MDPs called Opportunity Cost DEC-MDPs (OC-DEC-MDPs), in which interaction dependencies (in the form of precedence constraints) form a fully-connected, time-constrained dependency graph in which each agent's methods lie along a path through the graph. Temporal reasoning is then used to facilitate the formation of communication-free joint policies that coordinate the start times of agents' tasks.

# 4. THE CURSE OF DIMENSIONALITY

Becker gains traction by exploiting structure to compute solutions more efficiently. But while the complexity of DEC-MDPs with Event-Driven Interactions is reduced from that of the general class of DEC-MDPs, it is still doubly exponential in the number of dependencies[1]. And because the Coverage Set Algorithm (CSA) is an optimal solution method, its performance is critically affected by this exponential relationship. This curse of dimensionality manifests itself in two different ways in particular.

First, there is the incorporation of dependency information into local state. To maintain the Markov property, CSA requires that local models be augmented so as to include dependency history information, yielding an increase in the number of local states that is exponential in the number of dependencies. This increase can have a significant effect on the local policy computation, an operation that is employed repeatedly during the course of one run of CSA. Becker discusses intuitive characterizations that allow the augmented local MDPs to be kept small for his sample problems. In section 4.1, we further characterize these classes of problems, formalizing key structural properties that allow us to bound local state space complexity.

A second (and arguably more severe) impact is seen with the size of the CSA parameter space. As we will describe in Section 4.2, the number of parameters is no less than the number of dependencies, thus defining a many-dimensional space to navigate in order to compute the coverage set. In Section 4.2, we discuss the pitfalls of applying CSA as the number of dependencies scales up. We also relate this discussion to additional structural properties that could be exploited.

## 4.1 Exploiting Temporal Structure

As discussed in Section 3.1, Becker's algorithm breaks the DEC-MDP into augmented local models, making the state space exponentially larger in the number of interaction dependencies. To combat this increased complexity and render the coverage set algorithm tractable, Becker and colleagues focus their evaluation on TAEMS problems with *enablement* interactions. As it turns out, these problems contain special structure that give way to more efficient modeling methods. We formalize this additional event-driven dependency structure with the following definitions.

*Definition 1.* A DEC-MDP is **temporally indexed** if time is represented explicitly as a state feature, which we will call *time* (e.g. $time(s_0) = 0$).

*Definition 2.* A dependency $d_{ij}^k = \left\langle E_i^k, D_j^k \right\rangle$ is **temporally conditioned** if there exists a single **dependency time** $t$ such that:
$\forall s_j \in S_j, \left(\exists a_j | \langle s_j, a_j \rangle \in D_j^k\right) \Rightarrow (time(s_j) = t)$.

*Definition 3.* An interaction is a **temporally uncertain interdependency** if it may be represented by a set of temporally conditioned dependencies
$X_{ij} = \left\{d_{ij}^1, d_{ij}^2, ..., d_{ij}^k = \left\langle E_i^k, D_j^k \right\rangle, ...\right\}$ with the following properties:

[1]It is important to note that, given Becker's dependency semantics, there may be several dependencies required for each interaction (non-local effect). There is actually one dependency needed for each time that an interaction could occur.

1. $\forall d_{ij}^k \in X_{ij}$, $d_{ij}^k$ is *temporally conditioned* with the *dependency time* $t_k$
2. $\forall d_{ij}^x, d_{ij}^y \in X_{ij}, (t_x < t_y) \Rightarrow (E_i^x \subseteq E_i^y)$, which implies that $\forall s_j \in S_j$, if $d_{ij}^x$ is satisfied in $s_j$, so is $d_{ij}^y$
3. $\forall d_{ij}^x \in X_{ij}, \neg\exists d_{ij}^y \in X_{ij}$ for which $(t_x = t_y) \wedge \left(d_{ij}^x \neq d_{ij}^y\right)$

With these definitions, we have identified a type of interaction that is represented by the satisfaction of a temporally uncertain interdependency. Each of the temporally-conditioned dependencies in $X_{ij}$ represents a time point at which the temporally uncertain interdependency might be satisfied. As soon as one of the interaction dependencies is satisfied, all future dependencies are in turn satisfied. This is an intuitive property of *enablement* interactions: it may be uncertain when a task will get an enabled, but after that it can be successfully executed at all future time steps (subject to deadline constraints). This additional structure allows the dependency histories of all of the dependencies in $X_{ij}$ to be compactly represented in Becker's augmented local models. Only a single Boolean satisfaction variable $b$ need be added to the local state, representing whether the interdependency has been satisfied. The size of the local state space thus increases by a factor of just $2^{|X|}$ for DEC-MDPs with only temporally-uncertain event-driven interactions, whereas, in general, it increases by $T^{|d|}$ (because dependency histories are recorded).

*Definition 4.* For any given Event-Driven problem, the **agent interaction graph** summarizes agent influence, containing a node for each agent, and representing an edge between nodes $i$ and $j$ for every non-local effect that exists between agent $i$ and agent $j$ (or equivalently, for every way in which one can influence the other and vice versa).

THEOREM 1. *For a DEC-MDP with Event-Driven temporally uncertain interdependencies $\bar{X} = \left\{X^k\right\}$ where*

*(a) the only local state features observable by more than one agent are time and the interdependency satisfaction variable $b^k$ associated with each $\left\{X^k\right\}$*

*(b) there are no (undirected) cycles in the agent interaction graph,*

*For each variable $b^k$ modeled by agent $j$ and representing agent $i$'s satisfaction of $X_{ij}^k$, and for any policy of agent $i$, parameters $\left\{Prob(b^k|s_x), \forall s_x \in S_j\right\}$ can be completely summarized by $\left\{Prob(b^k|t), \forall t \in [0,T]\right\}$.*

PROOF SKETCH. In order to prove this theorem, it suffices to prove that there do not exist two of agent $j$'s local transition $\left\{s_x \rightarrow s'_x, s_y \rightarrow s'_y\right\}$ both occurring at time $t$ for which $Prob(b_k|s_x) = Prob(b_k|s_y) = 0$ and $Prob(b_k|s'_x) \neq Prob(b_k|s'_y)$. That is, given a fixed policy of agent $j$, there do not exist two trajectories of agent $i$ for which the time is the same but the probability of incoming satisfaction is different. Space limitation preclude a full proof, but let us provide some intuition as to why this should be true. If $\left(Prob(b^k|s_x) \neq Prob(b^k|s_y)\right)$, then there must be some differentiating feature (other than time) modeled by agent $j$ whose value is conditioned on the probability of agent $i$ satisfying $b^k$. In Event-Driven problems, the only dependencies between agent $i$'s local transitions and agent $j$'s local transitions are the event dependencies whose satisfaction is captured by dependency satisfaction variables. So one of these must be the differentiating feature. The two possibilities are (1) that agent $i$ has an outgoing dependency whose satisfaction differs in $s_x$ and $s_y$ that affect agent $i$'s transitions thereby affecting $Prob(b^k)$ or (2) that agent $i$ models another incoming dependency represented by $b'_k$ that is dependent on $b_k$ and whose satisfaction differs in $s_x$ and $s_y$. But the acyclicity of the agent interaction graph precludes both of these possibilities. So $Prob(b^k|s_x)$ must equal $Prob(b^k|s_y)$.

□

Theorem 1 has the implication that for certain problems with the assumed interaction structure dictated by conditions (a) and (b), the parameter space can be compacted significantly. Agent $j$ need only represent agent $i$'s policy with $T$ parameters per interaction whereas in the general case, it requires $|S_j|$ parameters per dependency.

## 4.2 Reducing the Viable Parameter Space

Even with the reductions implied by Theorem 1, the size of the parameter space of Becker's CSA increases significantly with the number of dependencies. In general, there is at least one parameter for each dependency. CSA finds the optimal joint policy by computing a coverage set, which is the set of all local policies of one agent that are a best response to any parameter value (that encodes the policy) of the other agent. This boils down to exploring all dimensions of the parameter set, computing a number of best-response policies that is (at least) exponential in the number of parameters. Thus, as the number of dependencies increases, the work that must be done by CSA increases exponentially.

We contend that although this parameter space is very large, temporally-uncertain interactions enable significant reductions in the number of parameter values that need be explored. CSA searches in a way that treats each parameter as an independent dimension. We propose to instead exploit the natural relationships that exist between the dependencies that make up a temporally-uncertain interaction. For example, if the probability of Agent 1 locating item X (from Figure 1) by time 3 is 0.5, the probability of it locating item X by time 4 must be at least 0.5. Intuitively, it makes sense to consider these parameters in combination instead of treating them as independent dimensions. The approach that we present in the next section takes advantage of these relationships to re-organize the parameter search space, mapping multiple related dependencies onto the same dimension.

# 5. COMMITMENTS

The DEC-MDP solution methods cited in Section 3 (by and large) have the flavor of policy search, involving computation of joint policies directly. We reformulate the DEC-MDP problem by decoupling the policy computation from the coordination of agent interactions. By allowing agents to negotiate over potential commitments to interactions, the problem is then turned into one of commitment-space search, whereby interactions are decided and then policies computed and evaluated around these interactions.

The idea of forming commitments to interactions and planning local behavior around those commitments has a rich history in the multiagent systems literature. Our work inherits ideas from Smith's *Contract Net* protocol for distributed processing[17] and Cohen and Levesque's theory of *Joint Intentions*[7]. Others have designed implementa-

tions of these commitment theories for solving multi-agent classical planning problems [9, 19, 23]. Here, we extend our previous work on MDP-based Commitment models [21, 22] so as to apply it to solving Event-Driven DEC-MDPs.

## 5.1 Commitment Semantics

We begin by adapting the commitment definition so that it corresponds to the satisfaction of a dependency:

*Definition 5.* A **commitment** $C\left(d_{ij}^k\right) = \rho$ to dependency $d_{ij}^k = \left\langle E_i^k, D_j^k \right\rangle$ is a guarantee that agent $i$ will adopt a policy such that event $E_i^k$ occurs in its execution (thereby satisfying $d_{ij}^k$ for agent $j$) with probability no less than $\rho$.

Commitments of this form allow agents to form promises to fulfill other agents' nonlocal dependencies. For example, our UAV agent from Figure 1 can promise to enable the UGV to execute its *obtain-item-X* task at time 2 with probability 2/3 by forming a commitment to the corresponding dependency. We further extend the commitment semantics to *temporally uncertain interdependencies.*

*Definition 6.* A **temporal commitment** $C\left(X_{ij}\right) = \langle \rho, t \rangle$ is a guarantee that agent $i$ will perform actions so as to, with probability no less than $\rho$, satisfy the temporally uncertain interdependency (represented by temporally-conditioned dependency set $X_{ij}$) by time $t$.

The *temporal commitment* is capable of representing more sweeping promises. By forming a temporal commitment, an agent is promising to satisfy all interaction dependencies whose times are greater than or equal to $t$. For example, this corresponds to the UAV (from Figure 1) promising to enable the UGV (with probability 2/3) to execute its *obtain-item-X* task at times 2, 3, 4, and 5.

## 5.2 Commitment Modeling

Commitments allow for compact representation of relevant nonlocal policy information. As with Becker's approach (as described in Section 3.1), this nonlocal information can be incorporated into the local state. With each temporally uncertain interdependency, we can associate a single variable $b \in \{true, false\}$ that corresponds to the satisfaction of the interdependency (as discussed at the end of Section 4.1). And just as in Becker's augmented DEC-MDP model, we can add this variable to the dependent agent's local state such that $b(s_i) = T$ implies that in state $s_i$, the interdependency has been satisfied.

Upon augmenting the local state as such, commitments must next be incorporated into the transition model. Given a temporal commitment to satisfy an interdependency by time $t$ with probability $\rho$, each transition $\langle s_i, a_i, s_i' \rangle$ leading into any state $s_i'$ with $time(s_i') = t$ must be expanded as follows:

$$\forall s_i', s_i, a_i | time(s_i' = t),$$
$$\begin{cases} P_i'(s_i', b = T | s_i, a_i, b = F) = P_i(s_i' | s_i, a_i, b = F) \cdot \rho \\ P_i'(s_i', b = F | s_i, a_i, b = F) = P_i(s_i' | s_i, a_i, b = F) \cdot (1 - \rho) \\ P_i'(s_i', b = T | s_i, a_i, b = T) = P_i(s_i' | s_i, a_i, b = T) \\ P_i'(s_i', b = F | s_i, a_i, b = T) = 0 \end{cases} \quad (1)$$

Because the interdependency is expected to be satisfied by time $t$ with probability $\rho$, we expect that $Prob(b|s_i) = \rho$ for all states at time $t$. The dependent agent thus models this satisfaction bit as being set at $t$ with probability $\rho$. All other transitions not influenced by some temporal commitment as above are governed by:

$$\forall s_i', s_i, s_a | time(s_i' \neq t),$$
$$\begin{cases} P_i'(s_i', b = T | s_i, a_i, b = F) = 0 \\ P_i'(s_i', b = F | s_i, a_i, b = F) = P_i(s_i' | s_i, a_i, b = F) \\ P_i'(s_i', b = T | s_i, a_i, b = T) = P_i(s_i' | s_i, a_i, b = T) \\ P_i'(s_i', b = F | s_i, a_i, b = T) = 0 \end{cases} \quad (2)$$

## 5.3 Commitment Enforcement

We can make use of past-developed Linear Programming methods [21] by which to compute an optimal local policy for a committing agent constrained so that its outgoing commitments are guaranteed. Constraints are added that represent the probabilistic fulfilment of commitment conditions. These commitment constraints can be trivially extended to enforce commitments to dependencies. Given a dependency $d_{ij}^k = \left\langle E_i^k, D_j^k \right\rangle$, consider the following linear programming constraint:

$$\sum_{(s_u, a, s_v) \in E_i^k} \left(x_{ua} \cdot P_i\left(s_v | s_u, a\right)\right) \geq \rho^k \quad (3)$$

Adding such a constraint to the standard MDP LP (as we have described in previous work [21]) directly constrains the occupancy measures $\bar{x}$ of the agent $i$'s policy, requiring that some event in $E_i^k$ occurs with probability at least $\rho k$. This corresponds exactly to semantics of a commitment $C_{ij}\left(d_{ij}^k\right) = \rho$. A *temporal commitment* $C_{ij}\left(X_{ij}\right) = \langle \rho, t \rangle$ can, in turn, be enforced with the following constraint:

$$\sum_{\left[E_i^k | \left(d_{ij}^k \in X_{ij} \wedge dt\left(d_{ij}^k\right) \leq t\right)\right]} \sum_{\left[(s_u, a, s_v) \in E_i^k\right]} \left(x_{ua} \cdot P_i\left(s_v | s_u, a\right)\right) \geq \rho^k \quad (4)$$

where $dt()$ corresponds to the *dependency time* characteristic from Definition 2.

## 5.4 Commitment-space Search

At the core of the commitment approach is the process of selecting the right set of commitments. The goal is to find commitments that, when modeled by the dependent agent and enforced by the supporting agent, produce high-quality joint policies that coordinate the agents' interactions. This has the flavor of the coverage set algorithm in that commitments provide a parametrization of the supporting agent's potential policies over which the dependent agent can search. More generally, commitments provide a parametrization of the interaction possibilities over which both agents can search. A (commitment) parameter setting for any given temporally-uncertain interaction takes the form of a time $t \in [0, T]$ and a probability $\rho \in [0, 1]$.

Increases in the temporal uncertainty of an interaction have little effect on this parametrization. The more potential times an interaction might occur, there a still just two dimensions (*time* and *probability*) over which to search for commitments. But with Becker's CSA, each additional time that an interaction could occur translates to an additional dependency and hence another dimension in the parameter space. We argue that, by mapping related dependency parameters onto different time values of same two-dimensional space, commitments enable a more scalable parametrization of temporally-uncertain interactions.

# 6. A COMPLETE COMMITMENT MODEL

A single temporal commitment conveys an accurate probability of interdependency satisfaction ($Prob(b|s_i)$) in states at time $t$ but not at times before or after $t$. We can complete the model by expressing temporal commitments at all potential satisfaction times:

*Definition 7.* A **complete temporal commitment set** $C^{complete}(X_{ij}) = \left\{C_{ij}^k(X_{ij})\right\} = \{\langle \rho_k, t_k \rangle\}$ is a guarantee that agent $i$ will perform actions so as to satisfy the temporally uncertain interdependency $X_{ij} = \left\{d_{ij}^1, d_{ij}^2, ..., d_{ij}^k, ...\right\}$ (with dependency times $\{t_1, t_2, ..., t_k, ...\}$) by each and every time $t_k$ with probability no less than $\rho_k$ respectively.

THEOREM 2. *For any Event-Driven DEC-MDP whose interactions are temporally uncertain interdependencies where*

*(a) the only local state features observable by more than one agent are time and the dependency satisfaction variables $\left\{b^k\right\}$, and*

*(b) there are no (undirected) cycles in the agent interaction graph,*

*there exists a configuration of complete temporal commitment sets (one per interaction) that, when represented using augmented local models (as in Section 5.2) and solved (as in Section 5.3) yield the optimal joint policy (equivalent to the one found by Becker).*

PROOF SKETCH. To prove this theorem, it suffices to prove that our commitment-augmented local model captures all of the state information that Becker's dependency-augmented local model does. From this it follows that since Becker's augmented local models are capable of representing local policies that together form the optimal joint policy of the DEC-MDP, so are our augmented local models.

Consider first the state information captured by Becker's augmented local model. Becker's augmented local state of agent $j$ adds only features that represent the execution history of each dependency $d_{ij}^k$. Since all dependencies are *temporally-conditioned* members of *temporally uncertain interdependencies* (each with a dependency time before and after which no state transitions are affected by the dependency's satisfaction), in any given state, it is redundant to record when such a dependency was satisfied. The only relevant information is whether or not the dependency was satisfied.

Due to property (2) of Definition 3, this information may be represented by an interdependency satisfaction variable $b$ and corresponding state information $Prob(b|s_i)$. A *complete temporal commitment set* explicitly represents $Prob(b|time = t)$ and combining this with the insight from Theorem 1, commitments and their respective models encode $Prob(b|s_i)$ for all local states. Therefore, subject to assumptions (a) and (b), commitment-augmented local MDPs capture all of the state information that Becker's dependency-augmented local models capture. □

Theorem 2 has the consequence that a commitment search methodology can, in theory, be used to optimally solve Event-Driven DEC-MDPs with temporally uncertain interdependencies subject to two additional structural assumptions. Since all of Becker's experimental problems also satisfy these assumptions, Theorem 2 applies to an interesting class of Event-Driven DEC-MDPs. Our future work includes extending the proof of Theorem 2 to a broader class of problems.

Theorem 2 proves that there must exist a set of commitments that, when enforced, will yield the optimal joint policy. Unfortunately, the space of possible temporal commitment sets is exponentially large, so finding the optimal set is intractable for large problems. Fortunately, as we have argued elsewhere [22], commitments provide a framework for tractable informed techniques that find useful approximate solutions.

# 7. FLEXIBLE APPROXIMATION

Sections 5 and 6 present two extremes of our commitment modeling approach. Section 5 associates a single *temporal commitment* with each interaction. For a dependent agent like the UGV from Figure 1, this single commitment provides an often crude approximation of the committed agent's behavior because the dependent agent only models a single time and probability. If an uncertain interaction occurs at one of the other possible times, the agents may miscoordinate. The benefit of such an approximation is compact models that may capture more concisely the most critical time and probability. Contrast this with modeling every time and probability as in Section 6. In general, this complete interaction model will increase the nonlocal information contained in the augmented local models, and thereby increase the computational complexity of computing local policies, finding the right set of commitments, and solving the problem. But if the right set of commitments is found, the agents will coordinate optimally.

Consider again the example shown in Figure 1. This problem has only one interaction that, given the fixed optimal policy of the UAV, could take place at time 1, time 2, or time 3 (depending on the probabilistic duration of *locate-X*). Performing an exhaustive commitment-space search would yield the following complete-commitment-set as being optimal: $\left\{\left\langle \frac{1}{3}, 1\right\rangle, \left\langle \frac{2}{3}, 2\right\rangle, \langle 1, 3\rangle\right\}$. When optimal local policies are formed around the commitments, the agents achieve the optimal expected utility of $3\frac{2}{3}$. At the other extreme, if the UGV were to represent its policy with only a single temporal commitment, the optimal commitment would be $\left\langle \frac{2}{3}, 2\right\rangle$, yielding an expected utility of $1 + \frac{2}{3} \cdot 3 = 3$. This is because the UGV is not modeling the case in which the UAV happens to locate $X$ at time 3. It is only modeling the potential interdependency satisfaction that occurs at time 2.

Which extreme is most suitable depends on the difficulty of the problem, the amount of planning time available to the agents, and the importance of high solution quality. A strength of using our commitment approach is that one need not pick either extreme. Our methodology is capable of representing a single commitment, all commitments, or a partial commitment set that flexibly balances quality and computation cost.

# 8. EMPIRICAL RESULTS

As a proof of concept, we have developed an algorithm that transitions from the single-commitment extreme to the complete-commitment-set extreme, computing better and better approximate solutions in an anytime fashion. This is a direct extension of our Commitment Negotiation algorithm [22], which greedily converges on a single time and

probability for each interaction through iterative negotiations between requester and provider. Our extension incrementally adds more commitments (one at a time) by greedily picking the best additional time and probability using the same greedy convergence protocol. Commitments are built up from just one per interaction to the full commitment set per interaction, and local models are gradually augmented with more precise information about each interaction.

In order to measure optimal solution quality, we have also implemented a centralized solution method that models the full DEC-MDP state space and uses linear programming to compute the optimal joint policy constrained so that agents base their individual policy decisions only on those features of their fully-observable local state.

## 8.1 Flexibility

We begin by applying our greedy incremental commitment-space search algorithm to a set of small problems to evaluate how much quality is lost due to approximation. Each problem contains two agents with three tasks each ($\{A_1, B_1, C_1\}$ and $\{A_2, B_2, C_2\}$). Each task has 3 different (equally-likely) durational outcomes {*minduration*, *minduration*+1, *minduration*+2}, where *minduration* is a randomly-selected integer in the interval $[1, 3]$. Each task is assigned a random integer quality selected uniformly from $[0, 3]$. All tasks have a global deadline of $T = 8$. There is a single enablement internal to each agent (between its local tasks): for agent $i$, an enablement constraint is added imposing $A_i$-*enables*-$B_i$ or $B_i$-*enables*-$C_i$ (with either case being equally likely). And between the two agents, there is one non-local enablement : $C_1$-*enables*-$A_2$. We have generated 25 of these random problems so as to create a diverse test set.

Figure 2 shows a bar graph of the quality achieved by the incremental greedy algorithm at each step (from commitment sets of size 0 to sets of size 8). Note that the for the first bar, no commitments are established. Instead each agent simply builds its local policy under the assumption that all incoming enablements will have probability 0 at all times. The Q* line is the quality achieved by the optimal DEC-MDP joint policy as computed by using the linear program described above. The results show that this greedy incremental commitment-space search method produces near-optimal quality, on average, for a diverse set of simple TAEMS problems. Shown alongside each quality bar is a hollowed-out runtime bar depicting the average computational cost of negotiating the corresponding commitment set. As more temporal commitments points are added, more computational resources are consumed, suggesting the flexible tradeoff of computation for solution quality that this incremental commitment-search method provides.

As a test of our claims from Theorem 2, we went on to run a naive exhaustive commitment-space search method on all 25 problems. Although it ran for substantially longer than the greedy commitment negotiation, our exhaustive search found a complete commitment set that yielded a joint policy with utility equal to that of the optimal joint policy for every problem.

## 8.2 Scalability

Next, we apply our commitment-space search algorithms to set of 25 larger problems. This time, each agent has 5 tasks, each with 5 possible durations and a global deadline of T=25. There is still only a single enablement interaction,

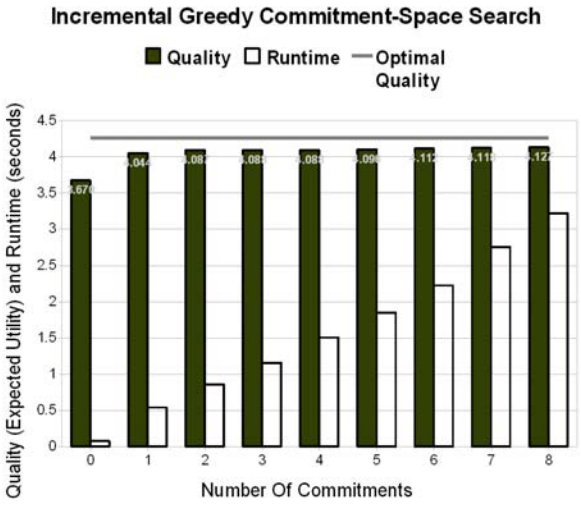


**Figure 2: Scalability Experiment**

but the greater degree of temporal uncertainty makes for much larger local models and complete commitment set with 25 temporal commitment points. A plot of the average solution quality of our greedy incremental commitment-space search (this time measuring runtime on the x-axis) is plotted in Figure 3. The dotted line underneath represents the average quality of the optimal 0-commitment solution. The algorithm converged on a complete temporal commitment set for each of the 25 runs in less than 90 seconds.

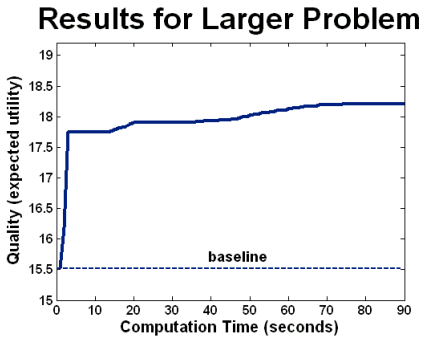


**Figure 3: Incremental Greedy Commitment-Space Search Quality**

Due to the large degree of uncertainty inherent in these problems, each interaction, if modeled with Becker's CSA algorithm, would need to include 25 dependencies, yielding a 25-dimensional parameter space. Navigating a space of such high dimensionality is a daunting task, and intractable to search completely. Commitment negotiation avoids this high dimensionality by approximating the parameter space, attributing just two dimensions to the interaction: a time and a probability, and searching in an informed manner. This experiment, though preliminary, suggests the scalability of the commitment solution methodology in computing near-optimal approximate solutions.

# 9. CONCLUSION

The main contribution of this paper is a reformulation of the Event-Driven DEC-MDP problem into one that can be solved with commitment-space search algorithms. In pursuit of this goal, we have identified further structural properties of certain Event-Driven problems and developed theory to show how these properties can be exploited. We have developed a commitment framework that exploits temporally uncertain structure, proving (under some restrictions) that this can produce optimal joint policies for Event-Driven DEC-MDPs equivalent to Becker's optimal algorithm. We have also demonstrated that our methodology can represent a range of approximate solutions when planning time is limited. Because it avoids the dimensionality explosion of optimal solution methods like CSA, it can scale to provide useful approximate coordination solutions to larger problems than existing solutions methods.

We have provided preliminary empirical results in support of our claims, developing an anytime commitment algorithm that scales well in providing coordinated, approximate solutions to problems with large amounts of temporal uncertainty. But further empirical investigation is needed to determine how commitment-driven approximate solutions compare to approximate solutions computable with other existing methods like CSA for problems small and large.

## 10. ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Office of Scientific Research under Contract No. FA9550-07-1-0262.

## 11. REFERENCES

[1] C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *Proceedings of the Twenty Third Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[2] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 302–309, Washington, DC, USA, 2004. IEEE Computer Society.

[3] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[4] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[5] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *IJCAI*, pages 1287–1292, 2005.

[6] A. Beynier and A.-I. Mouaddib. A polynomial algorithm for decentralized Markov decision processes with temporal constraints. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 963–969, New York, NY, USA, 2005. ACM.

[7] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[8] K. Decker. TAEMS: A framework for environment centered analysis & design of coordination mechanisms. In *Foundations of Distributed Artificial Intelligence, Chapter 16*, pages 429–448. G. O'Hare and N. Jennings (eds.), Wiley Inter-Science, 1996.

[9] E. H. Durfee and V. R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991.

[10] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[11] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games, 2004.

[12] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Eighteenth national conference on Artificial intelligence*, pages 468–475, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[13] J. Marecki and M. Tambe. On opportunistic techniques for solving decentralized Markov decision processes with temporal constraints. In *AAMAS-07*, Honolulu, HI, 2007.

[14] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI05*, pages 133–139, 2005.

[15] M. Petrik and S. Zilberstein. Anytime coordination using separable bilinear programs. In *AAAI*, pages 750–755, 2007.

[16] S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *IJCAI*, pages 2009–2015, 2007.

[17] R. G. Smith. The contract net protocol: High level communication and control in distributed problem solver. *IEEE Trans. on Computers*, C-29(12):1104–1113, 1980.

[18] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 576–590, 2005.

[19] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83, 1997.

[20] T. Wagner. The DARPA/IPTO COORDINATORS Program, 2004. [www.darpa.mil/IPTO/Programs/coordinators/].

[21] S. Witwicki and E. Durfee. Commitment-driven distributed joint policy search. In *Proceedings of the Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pages 480–487, Honolulu, HI, 2007.

[22] S. Witwicki and E. Durfee. Commitment-based service coordination. *Int. Journal of Agent-Oriented Software Engineering*, 3(1):59–87, 2009. [www.umich.edu/~witwicki/pub/IJAOSE.pdf].

[23] P. Xuan and V. R. Lesser. Incorporating uncertainty in agent commitments. In *Proceedings of ATAL-99*, pages 221–234. Springer-Verlag, 1999.

# Communication, Credibility and Negotiation Using a Cognitive Hierarchy Model

Michael Wunder
Rutgers University
mwunder@cs.rutgers.edu

Michael Littman
Rutgers University
mlittman@cs.rutgers.edu

Matthew Stone
Rutgers University
matthew.stone@rutgers.edu

## ABSTRACT

The cognitive hierarchy model is an approach to decision making in multi-agent interactions motivated by laboratory studies of people. It bases decisions on empirical assumptions about agents' likely play and agents' limited abilities to second-guess their opponents. It is attractive as a model of human reasoning in economic settings, and has proved successful in designing agents that perform effectively in interactions not only with similar strategies but also with sophisticated agents, with simpler computer programs, and with people. In this paper, we explore the qualitative structure of iterating best response solutions in two repeated games, one without messages and the other including communication in the form of non-binding promises and threats. Once the model anticipates interacting with sufficiently sophisticated agents with a sufficiently high probability, reasoning leads to policies that disclose intentions truthfully, and expect credibility from the agents they interact with, even as those policies act aggressively to discover and exploit other agents' weaknesses and idiosyncrasies. Non-binding communication improves overall agent performance in our experiments.

## Categories and Subject Descriptors

F.1.1 [**Computation by Abstract Devices**]: Finite Automata; G.3 [**Probability and Statistics**]: Markov Process; I.2.0 [**Artificial Intelligence**]: Cognitive Simulation; I.2.11 [**Artificial Intelligence**]: Distributed Artifical Intelligence—*Intelligent Agents, Multiagent Systems*

## General Terms

POMDPs, Cognitive Hierarchy, Repeated Games

## Keywords

Agent Communication, Game Theory, Multiagent Learning

## 1. INTRODUCTION

When agents repeatedly face the same simple strategic problems, we can expect them to exhibit *equilibrium* behavior—where each acts to optimize his outcome given how the others will act; see Leyton-Brown and Shoham [8] for a computational introduction. For rational agents, such settings offer the information and computation needed to deliberately work out a strategy that takes others' choices into account. But even agents that act by trial and error will reach equilibria in such settings, through the reinforcement provided by patterns of past play [10]. Indeed, in repeated simple strategic interactions, we do find equilibrium behavior in people and many other organisms.

It is different for situations which do not obviously call for strategic reasoning, for situations which are novel or complex. Here, it becomes unlikely that agents will enjoy the information, computation or history that make equilibrium play feasible. Instead, we expect agents to exhibit heuristic, approximate or biased solutions to the problems of interaction [1, 9]. Think of a a lovers' quarrel, a game of chess, or a corporate acquisition: these are interactions people muddle through, not interactions they solve. In interesting domains, agents will have to go beyond first-principles reasoning about interaction to model the heuristic basis of their partners' decisions: the gut reactions, the resource limits, and the background culture of practices and expectations that actually lead to choices. Agents may also have to work explicitly to create interactions where their own and their partners' limited reasoning is effective. Such thinking is the mainstay of human social life: not only in the trust, rapport and relationship we can achieve in alignment with one another, but also in the circumspection, prudence and vigilance we otherwise maintain.

This paper presents a set of simulation results designed to explore the diverse reasoning that might be required for interacting agents in realistic situations. Our basic tack is to approach interaction not as strategy but as a more constrained planning problem for sequential action under uncertainty. In particular, we use the computational framework of partially-observable Markov decision processes (POMDPs [6]) as a common substrate for our agents' reasoning. Our POMDP framework captures the key features of interactive reasoning by simulating the fixed decision making of relevant possible opponents. We can represent *background expectations* and *heuristic choices* by simulating opponents that meet specific expecations, or follow specific heuristics. We can characterize *limited strategic reasoning* through a cognitive hierarchy model—simulating more sophisticated opponents that tune their behavior to simpler opponents [3]. And finally, we can investigate *adaptive response*, because the POMDP framework describes the tradeoff between acting to achieve immediate payoffs and acting to reveal qualities of opposing players that can be exploited in further

AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2009, Budapest, Hungary.

decision making.

The method we introduce applies POMDP solving techniques to find the best response to a given population of strategies. A new population is built by combining the old one with the new computed strategy. As this process is repeated and the population grows according to a predefined Poisson distribution, earlier strategies recede in importance somewhat. In this way, our proposed algorithm repeatedly computes new strategies and attains more experience against more sophisticated agents, as it converges toward some policy that will yield the same policy in response. This procedure therefore determines a *cognitive hierarchy model*, as explained more fully in the Section 2.1.

We work specifically with the classic prisoner's dilemma game, iterated for a small number of rounds with an opponent with limited, heuristic, or biased decision making. In this setting, our agent will do best by defecting against uncooperative or credulous opponents, and by engaging with reciprocating opponents who respond to cooperation with cooperation but defection with defection. Thus, our agent must learn how its opponent plays, and signal how it plays, as it accumulates payoffs in initial rounds. We sometimes give agents the opportunity to explain themselves: to offer non-binding promises and threats describing the expected course of future interactions. Our simulation results show that agents in this general setting can achieve collaboration in some cases, but that they face a difficult problem in establishing trust with a reciprocating opponent while guarding against exploitation by uncooperative opponents. Communication can help, by allowing agents to establish trust by truthfully describing harmful actions they must take to avoid leaving themselves vulnerable to exploitation. Specifically, once the model anticipates interacting with sufficiently sophisticated agents with a sufficiently high probability, reasoning leads to policies that disclose intentions truthfully, and expect credibility from the agents they interact with, even as those policies act aggressively to discover and exploit other agents' weaknesses and idiosyncrasies. The results offer a suggestive link between the challenges agents face in connecting with one another and the constructs that people—or agents—can use to pursue those connections.

# 2. RESEARCH SETTING

Traditional learning methods find it difficult to learn in games where other agents are also learning. The algorithms often require a static environment and the presence of other learning agents invalidates this assumption. One commonly cited example is the rock–paper–scissors game, where each one of the three moves dominates one other. In this case, crudely adapting agents will circle each other indefinitely, as each reacts to a model of the other that is out-of-date.

One lesson is that acting effectively amid other learners requires a new way to generalize from past experience. It is not enough to assume that other agents will act as they have in the past. Others may think strategically, and arrive at behavior that differs from anything that they have done so far. To prepare for this, agents must second-guess one another, and attempt to model their opponents' possibly creative decision making. Such reasoning strategies can lead to good performance against learning agents, even in problematic cases like rock–paper–scissors [5].

In general, we could approach this second-guessing through the symmetric strategizing of game theory [8], through open-ended general models of agents' reasoning [5], or through a range of further simplifying assumptions. The cognitive hierarchy model offers one such simplification; it offers a framework for bounding strategic reasoning by assuming that agents have heuristic limits on the degree to which they are willing to second-guess one another [3]. The model is intuitively appealing, because it captures the empirical observation that players are boundedly rational, with limited amounts of working memory, and furthermore believe that others also have bounds on their rationality. Moreover, as we shall see, it has a straightforward implementation using off-the-shelf planning techniques.

## 2.1 The Cognitive Hierarchy Model in Games

A cognitive hierarchy model is defined in terms of a set of base policies and a series of levels of sophistication describing agents' possible strategic reasoning. The base policies involve no strategic reasoning; they simply prescribe a distribution over actions to perform in each state. The least sophisticated agents in a cognitive hierarchy, the 0-step agents, are those that directly follow one of the base policies. We assume an initial distribution over the 0-step agents.

More sophisticated agents at level $k+1$ in the hierarchy perform best-response reasoning, on the assumption that other agents are drawn from a distribution of agents at level $k$ or lower. Thus, agents at level $k+1$ have no model of players doing more than $k$ steps of thinking. For example, a 1-step agent second-guesses the 0-step agents by planning a best-response to their distribution of strategies.

In general, $k$-step players must plan their responses based on a distribution over the sophistication of their opponents. A simple assumption is that their information about less sophisticated agents is correct; their "mistake" is to ignore the possibility of more sophisticated agents. Formally, if the actual proportion of $h$-step players is $f(h)$, a $k$-step player will believe the proportion of $h$-step players to be $g_k(h)$, where $g_k(h) = 0 \ \forall \ h \geq k$ and $g_k(h) \propto f(h)$ otherwise. Finally, as $k$ increases, we assume that $k$-step players become exponentially less and less likely, so this estimate converges to a fixed distribution. Essentially, the difference between players doing $k$ and $k+1$ steps of thinking will shrink as $k$ increases and the proportion of agents reasoning more than $k$ steps get smaller. Under these conditions, the players should converge towards some strategy. However, we might not expect this algorithm to end up at a Nash equilibrium. The agent must respond to the original naive strategies as well as the more sophisticated ones, as a result of keeping some of them in its model of the environment.

The distribution $f(k)$ is the main component of the cognitive hierarchy model, and is determined by the parameter $\tau$ corresponding to the average $k$ in the population. Camerer et al. [3] argue that $f(k)/f(k-1)$ is proportional to $1/k$, resulting in $f(k)$ as the Poisson distribution, so that $f(k) = e^{-\tau}\tau^k/k!$. This distribution has a number of advantages: it is simple to compute as it has only one free parameter, $\tau$, and it closely fits empirical observations.

One example where this model is useful is the Keynesian beauty contest [7] where players score highest by picking the contestant that others value most. In the $p$-beauty contest, numbers are substituted for beauty contestants. Each of $N$ players chooses a number $0 \leq x_i \leq 100$. The average of these numbers scaled by $p$ specifies the winning number. The player who chose closest to the winning number wins a

fixed prize. The Nash equilibrium for this game is for everyone to choose 0 when $p < 1$, but people do not reach this equilibrium immediately. A cognitive hierarchy model can explain these results in terms of players' expectations over how deep the average reasoning will be, with increasingly sophisticated agents reaching increasingly low estimates of others' likely bids.

## 2.2 POMDPs

Partially observable Markov decision processes, or POMDPs, are a mathematical model for decision making under uncertainty and have been used productively to define and solve for optimal policies for agents [6]. A POMDP model consists of sets of states, actions, observations, and probabilistic functions relating them.

The problem of computing a policy that plays optimally against a finite-state opponent chosen from a finite set can be cast as a POMDP, consisting of an underlying MDP $< \mathcal{S}, \mathcal{A}, T, R >$ and set of observations $\mathcal{O}$. Here, the states $\mathcal{S}$ of the POMDP are the states within the possible strategies. The actions $\mathcal{A}$ are the choices of the agent and transition function $T$ is determined by the opponent strategy. Rewards $R$ are a property of the game and depend on both players' actions. The observations $\mathcal{O}$ are the opponent's observed responses. When there is communication in the game, the receipt of messages between rounds map to additional observations in separate states. An agent starts out in state $s$ against strategy $i$, but this state is unknown to the agent.

Through repeated interaction over the length of a POMDP, an agent computes its belief state $b$, which is a cross product of all possible strategies and the current state in each such strategy. As the agent traverses the graph defined by the opponent's strategy, it achieves greater certainty about the identity of that opponent as well as the proper response. In other words, each new observation $o$ brings $b$ more in line with the correct state within the actual strategy of the opponent. Solving methods balance the value of this information with well-rewarded actions based on $b$.

A POMDP solver available on the web (www.pomdp.org) presents a straightforward protocol for specifying a POMDP, including states, transitions, actions, rewards, and observations [4]. In combination, these five inputs constitute a POMDP. The solver uses several state-of-the-art exact algorithms to calculate the optimal policy and values for each state depending on the actual probability for these states. If the solver can construct an optimal policy within reasonable time bounds, it outputs this strategy the form of a finite automaton, which is easily incorporated as another entry in the cognitive hierarchy for future optimizations.

To capture a cognitive hierarchy with a POMDP solver, we run the solver on a model that describes interaction with a distribution of 0-step agents. The solution describes a 1-step agent. From here on out, whenever we derive a solution agent for step $k$, we can add the agent to the distribution of agents with weight given by $h(k)$, and repeat the solution process, yielding the automaton for step $k + 1$. The procedure can be iterated indefinitely.

# 3. TRADEOFFS FOR REASONING

Games with cooperative outcomes and an incentive to cheat, like the prisoner's dilemma, are good settings for exploring ways to encourage and enforce cooperation between agents with conflicting interests. We consider solutions to finite iterated prisoner's dilemma problems under a cognitive hierarchy model. We begin with a range of base policies, some of which reward cooperation and punish defection, and some of which do not.

The best response against these base agents amounts to a tradeoff. Defecting offers an opportunity to discover what kind of opponent one faces, and perhaps to exploit them. But it forgoes opportunities to cooperate with reciprocating agents, against which cooperation is the best outcome.

More sophisticated agents eventually face different tradeoffs. They must not only be able to size up base agents, but they must be able to recognize and interact appropriately with agents that carry out limited strategic reasoning. Depending on the prevalence of the different levels of reasoning in the environment, a more sophisticated agent may be forced to surreptitiously masquerade as a reciprocating base agent in order to elicit good behavior from less sophisticated agents.

## 3.1 Communication and credibility

Consider what happens if agents can exchange nonbinding threats and promises that do not impact the payoffs of the game. In equilibrium analyses, 'cheap talk' of this form has an effect on the outcome of a game only under certain constraints. In particular, the type of the sender agent must make a difference to the receiver, and the sender and receiver cannot diverge in preferences. While for us the first condition is clearly met, the second is not because the agents have incentive to defect from the cooperative outcome, and therefore to *deceive*.

In the cognitive hierarchy model, however, the effect of cheap talk depends not only on the game itself, but also on the behavior of the base agents and the tradeoffs made by agents of different levels of sophistication. We include reciprocating agents that describe their future intentions honestly. Meaningful communication cannot occur without some agent in the population who engages in enough truth telling to make it worthwhile to believe the agent.

Best responders to these base strategies then have the option of demanding integrity rather than demanding reciprication, or in addition to it. Such agents provide benefits to truth-tellers, and punish deceivers.

More sophisticated agents may not only call for integrity but realize that they must behave with honesty if they are to cooperate with the less sophisticated agents that demand integrity—the analogue of masquerading as the kind of base agent that these agents cooperate with. In the end, such agents communicate successfully, truthfully sharing the information that they are hard-nosed players that nevertheless intend to reciprocate with one another. This is a way to get meaningful communication despite the nonaligned interests of the communicators.

## 3.2 Learning and teaching

Reasoning in the cognitive hierarchy contains elements of strategic teaching [2], as a side-effect of the iterated planning involved in solving POMDPs. In essence, by computing the optimal series of actions given a certain 0-step distribution over opponents, the output 1-step solution will act to help train the 2-step generation's computed policy. The next policy shaped in part by the interactions of the previous one will be guided towards the 0-step reciprocal strategies. This effect will feed on itself, and the 2-step agents will in turn

guide the 3-step generation even more strongly. Another property of the algorithm based on the cognitive hierarchy viewpoint is that the proportion of sophisticated agents that each new $k$-step POMDP contains gradually rises for all values of $\tau$, leading to a situation where the majority of agents contain some level of sophistication against several classes of opponents.

Other research has looked into adaptive methods to finding best actions, notably experience-weighted attraction (EWA), which itself blends two other techniques, relying on reinforcement and belief updating [2]. Our iterated method attempts to compute an exact policy for the entire game period and finds the optimal series of moves for the total payoff. Another difference is that EWA tries to adapt to changing players as the game progresses. This algorithm yields stationary strategies and the contribution of each strategy to an agent's model changes as $k$ and $\tau$ increase. Our model considers other strategies to be stationary multi-step decision rules and must adapt to changes in the relative proportions of such rules.

# 4. EXPERIMENTS AND RESULTS

The following experiments were run with Cassandra's POMDP solver from www.pomdp.org [4]. The game is a six-turn version of prisoner's dilemma, with the following payoff matrix for player 1. Player 1's actions are on rows, player 2 on the columns. C stands for the cooperate action and D for defect:

$$\begin{bmatrix} P1 & C & D \\ C & 3 & 0 \\ D & 4 & 1 \end{bmatrix}$$

In this game, a player is always better off choosing D, which leads (D,D) to be known as the Nash equilibrium for this game. However, these values tend to create cooperative behavior when playing is repeated. There is also a discount factor of 0.95 for each successive state. Where communication is present, the message-exchange step is a separate state, but for purposes of comparing relative scores the discount will be excluded.

One useful feature about this solver that is convenient for these experiments involves the output values for each belief state. This data is output along with the policy graph finite automaton solving the POMDP. While the solver provides one policy based on the starting state specified in the program, policies for different starting belief states can be calculated without running the solver again. This way, the policies can be quickly recomputed without creating new files for each desired combination of belief states.

## 4.1 Iterated Prisoner's Dilemma Without Communication

Imagine that you are told to play IPD for six turns against a single player selected at random from a population. The majority of this population will be unknown. Some, but not all, of your possible opponents will consist of the following ratio of four strategies, defined as the 0-level. Three of the four 0-level strategies are non-reciprocating, including a strategy that always cooperates, one that always defects, and one that simply picks an action randomly, with each action equally likely. A fourth strategy is reciprocating, meaning that it pays to cooperate with it as cooperative actions will be rewarded with cooperation, while defections will tend to be punished with defections. A player has equal chance to meet each of these strategies. We might consider them to represent simplified personalities that exist in the world. There exist the altruist, the mean individual, the crazy, random strategy and the reasonable, reciprocating player. The ratio of three to one is derived from the relative costs of defecting against the two classes of agents. On average, the payoff of defecting against a non-responsive player is +1, while the average penalty for defection against a fully reciprocating agent is no smaller than -1.5 in the long run (-3 every other turn). A minor alteration is that a small percentage around 20% of the reciprocating population starts off defecting while the rest cooperate. While this change does not have much effect on the actual policies, it makes exploration easier.

Examples of reciprocating agents include Tit-for-Tat (TFT) and Pavlov. TFT starts by cooperating and then simply returns the action of the other player from the previous round. After the first round, Pavlov will cooperate if the players play the same action in the previous round, and otherwise it defects. For the reciprocating player we have chosen to use the Pavlov strategy, for several reasons. First, Pavlov achieves a subgame equilibrium in self-play in repeated settings. It has a credible deterrent and is somewhat more robust than TFT where some degree of noise is present, as two TFT agents can easily start alternating defections. It is also an evolutionarily stable strategy. Another reason is as there are only a short number of rounds of play available, it is necessary for a strategy to achieve coordination quickly. TFT, while ideal as a deterrent, adds uncertainty when players find themselves in mutual defection. A player will have to cooperate for two more turns before finding out whether the opposing strategy matches TFT, while against Pavlov it will be known the following turn.

It is uncertain what a player should do in a situation like this one, where players other than these given strategies will be part of the population alongside the known participants. These other players should be considered to have some reasoning capacity, and are trying to figure out what to do also. Someone might start by saying that it would be nice to cooperate with the reciprocating player, assuming that the same actions are observed. However, the observed actions are the same for the always C agent, so it would also be nice to know which one we are facing, as there is substantial reward for having this knowledge. A player can only find out this information by testing the other player with a defection move. The problem comes when the player starts thinking past this point to how more advanced players will view the game. If a defection is sent out the first round, it will be indistinguishable from the constantly defecting agent. So, perhaps it would be better to wait a certain number of turns, but during this waiting period a player is giving up valuable opportunities. It would appear that this type of game is suited for cognitive hierarchy analysis, where the four given types of player constitute the 0-step thinker in that model.

In order to implement our version of the cognitive hierarchy model, we include each $k$-step agent according to the predefined Poisson distribution. The primary parameter for the distribution $f(k)$ is $\tau$, and it will be varied across trials, also determining the actual number of 0-step players. Once a policy against the initial distribution is computed, it is called the 1-step policy. It is included in the set of strate-

gies in the proportion given by $f(1)$ for the second iteration of POMDP, which will yield the 2-step agent. However, the estimated population does not yet include agents with step greater than 1, so at every stage only policies computed with step size $<$ k will be present. The distribution of strategies in round $k$ will be attained by renormalizing the contribution of the original strategies given by $f(0)/\sum_{i=0}^{k-1} f(i)$ . The process is then repeated to find the 3-step policy, and so on.

Some simple calculation will show that even with low levels of reciprocating agents, it is worthwhile to engage in cooperation, provided that a player can become fairly confident about the opponent after several testing rounds. Figure 1 shows the number of testing moves required to distinguish a reciprocator from a completely random agent, over varying prior ratios for the reciprocating agent. If the agent fails the test, the other agent is clearly random and exploitable.

When Pavlov makes up 25% of the initial arrangement, the first computed policy begins by testing the field with a single defection, and then follows the rule set by Pavlov. Namely, if the other agent has defected also, cooperate next. Otherwise, defect again. This policy can be seen in Figure 2. This combination accomplishes two things: it forces the reciprocating agent to respond with a punishing defection on the following round, and it reveals some of the non-responsive agents such as the always cooperating agent. Since there is only one correct sequence of actions for this test, the random agent gives itself away with increasing likelihood. If the opposing agent passes the test, the 1-step player is then safe to pursue mutual cooperation at least until the final round, when there is no consequence for taking the more profitable defection move.

This strategy is placed into a new POMDP at a proportion derived from the current $\tau$. The 2-step policy is then computed. See Figures 2-5 for the first four steps of reasoning computed in this manner, with $\tau \leq 2$. To read the finite automata graphs, start at the state with the double circle. The action inside the circle, whether C or D, describes the agent's action. The arrows exiting the circle are the actions played by the player facing this agent. The agent behavior can then be tracked until the game ends.

Another feature of the policies with small $k$ is that unwarranted defections, such as those that occur without a prompting test defection, are classified as non-responsive. This feature has consequences for the next round of policy computation, because the condition of one or more initial cooperation moves must be met to warrant future cooperation from the population of 1-step policies. As Figure 3 shows, the 2-step policy finds it beneficial to wait a turn before testing the opposing agent, as well as to watch for the pattern defined by the reciprocating 1-step agent. For lower values of $\tau$ this process continues until $k$ is about 4, where the number of previous $(k-1)$-step agents are too small to consider. The final policy of this particular sequence for $\tau \leq 2$ in Figure 5 shows how waiting three turns is the best policy, followed by constant defection if it has not been tested by that point.

For intermediate $\tau > 2$ , the process gives a strategy where agents find that it is only worth cooperating for a single round if the other agent is very forgiving, due to the wide range of strategies present in the population. In the highest values of $\tau$ measured, about 3.5 and 4, a new policy arises in response to the dominant one, which acts very obediently in order to receive its one cooperation outcome from the previous strategy.

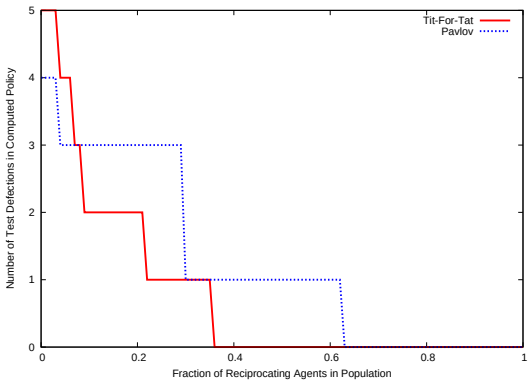


**Figure 1: Number of testing rounds in a game between random agent and reciproctor required to reduce probability that the opponent is random. Plotted over the true probability of reciprocation.**

Figure 6 demonstrates the relationship between $\tau$ and the overall population's score. We see a slight rise for low values of $\tau$, with a peak around 2 followed by a steep dropoff. In general players do better within a population by reasoning more, but the population suffers when too much reasoning is done because the cooperation unravels from the final turns.

## 4.2 Iterated Prisoner's Dilemma With Communication

The same method applied to games with communication proceeds somewhat differently. Here, the players are able to send a message before odd-numbered turns, which consists of a two-bit string. This message is meant to correspond to the next turn consequence to the opponent for its choice of action in the current round. That is, the first string describes the sender's action resulting from a receiver's cooperate action, and the second bit of the string describes the response to the receiver's defection action. Thus, two rounds are required between messages. This conditional structure can also be applied to games with with more than two actions.

These messages will gain their meaning from the agents that intend to enforce the truth value of messages they send, even if only consisting of a minority of the total population. An important point concerning these messages is that only two of them actually contain evidence that the sender will reciprocate. A message containing the same action for both conditions says either that the receiver's action will be ignored in the decision for the next turn, or that the sender does not mean what it is saying. In either case, these messages express that the sender is an unreliable communicator and does not deserve reciprocation. This condition holds as long as there are no reciprocating agents that fail to communicate correctly.

The message stage opens up more possiblities for the initial 0-step agents to play. There are four new actions between every second round that are seemingly unrelated to the underlying strategy. The communication strategies mirror their corresponding actions. That is, the virtuous reciprocator agent follows a Pavlov strategy that communi-

cates that strategy truthfully. The random agent sends random messages. The all-cooperator and all-defector attempt to blend in with the truthful population by restricting the messages to the Tit-for-Tat message, which states that an action will be met with the same action on the following turn. Looking more closely, the naive cooperator demonstrates even greater naivete in the hope that an empty threat will induce cooperation. Since this message is the one most commonly used by the Pavlov strategy, it also makes sense that an all-out defector would send it, as this lie has the biggest chance of success. In general, the games with and without communication were made as close as possible in the sense that the same starting population was used in both. While communication is itself a big difference, it is not immediately obvious that it will have much impact on the actual results.

The population of $k$-step agents follows the same distribution as the non-communication case. When $\tau$ is varied, the fraction of each policy changes. As the policy computation is repeated, initial differences in the distribution over $k$ will lead to different policies, and in substantial divergence in later rounds.

In the communicating population, the 1-step policy computed on this 0-step distribution defects right away on the non-information messages, as only the random agent sends such messages. It also becomes very sensitive to the truth content of the messages. Senders that do not conform to their own messages by following through on their promised consequences are weeded out by the second turn. To complete the evaluation, the agent starts off by testing the results of defecting, and if the action is unpunished continues that exploitation. Otherwise, reciprocation continues pending further adherence to the truth contained in the messages. It is important to note, however, that this policy does not meet its own standards. Since there is no opposing agent that will exact a consequence for reneging on promises, it does not pay attention to the messages it sends.

The 2-step agent does send meaningful messages, and sticks to them until an opposing promise is broken. The presence of the 1-step agent is enough to select for this property. Furthermore, because the computed policies find truth enforcement to be a better indicator for future reciprocation, there is no need to seek the cooperative action from the beginning. Therefore, the 2-step agent is free to test by defecting right away, and checks that the other agent will follow direction by sending a specific message that corresponds to the message a Pavlov agent would send in that scenario. Specifically, the message states that "'I will defect next if you cooperate this turn, and vice versa"'. Any logical agent could not resist this temptation to defect, and so if the other agent matches this behavior, mutual cooperation can begin on the following turn. If the other agent starts off with the Tit-for-Tat message, the correct sequence will be cooperate/defect for the opposer, and defect/defect for the computed 2-step policy. Mutual cooperation can then ensue in the third turn, as long as the messages now contain the Tit-for-Tat instruction.

An agent dealing with the 2-step agent will be led to truthful communication by the cascade of reasoning suggested in Section 3.1. The 2-step agent finds the cooperative outcome with the Pavlov agent as well as other sophisticated strategies. Second, it tests the other agent's messages for accuracy. Third, it enforces this credibility test with a penalty of endless defection for failure. Finally, it tells the truth itself, at least until the last round when future actions cease to matter.

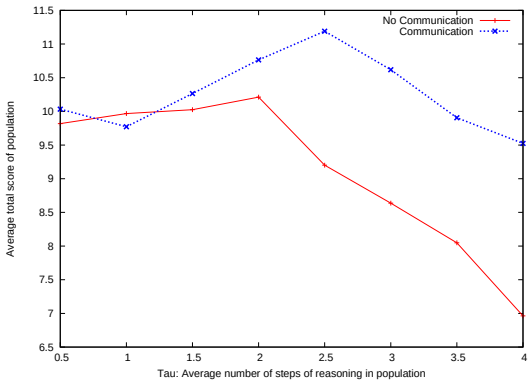


**Figure 6: The total score of a population of agents, weighted by proportion of population, as the average level of reasoning changes**

Across the board, as $\tau$ varies, this same pattern is observed. Although more advanced agents begin defecting earlier, the initial moves remain the same. The result of this behavior is that future iterations of agents will choose initally matching strategies, instituting some degree of coordination. The benefits of this outcome are obvious. Instead of several turns of enforced waiting as with the non-communicating agents, the agents are able to test their opponent immediately upon commencing the game, and so separate with a high degree of confidence the reciprocating agents from the non-responsive ones. In addition, as the number of strategies remains small, each new policy will not have to change much and the coordinated outcome is enforced. Note that this outcome is not explicitly devised, as in other signaling games where coordination is obviously desirable. Instead, it arises from the decentralized discovery that communication can work to bring players to a better space.

The experimental results confirm that the communicating agents following this strategy achieve higher scores over the total run of the game. Communication is also better for the scores when high reasoning ability is present, both within and between populations. Figure 6 shows the increase in performance for communicating agents as $\tau$ rises. As $\tau$ increases, the scores begin to diverge from those non-communicating games until the gap between them is over 2.5 points. The performance peaks at a higher reasoning level, and degrades more slowly as reasoning rises. This amount understates the true advantage in scores, because in reality the variance across strategies is much less than the maximum of 24. One of the main reasons for an improvement with communication is that the higher $k$-step non-communicating agents spend time waiting, which hurts performance against the exploitable members of the population. Communicators have the ability to side-step the waiting issue by directly stating what they are planning to do.

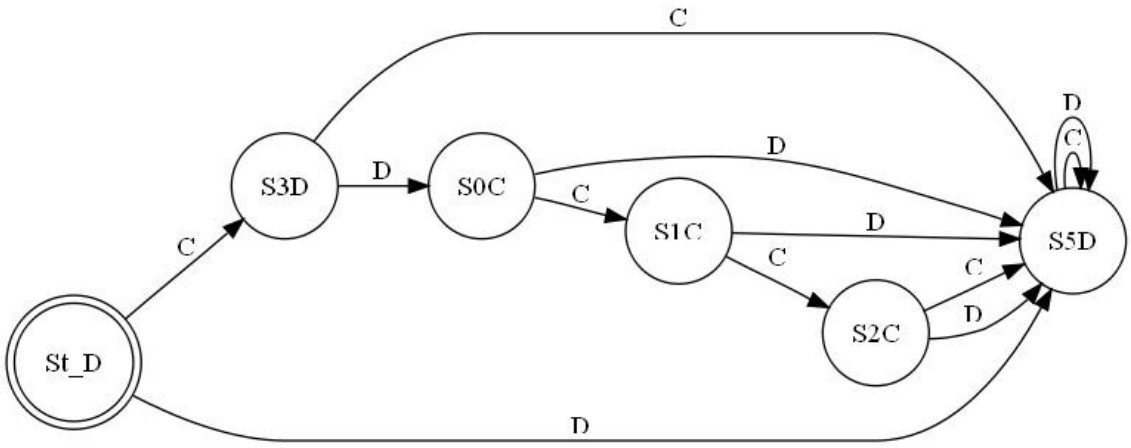


**Figure 2: An agent in the six-turn IPD that does not wait before test defecting. $\tau \leq 2$, k = 1.**

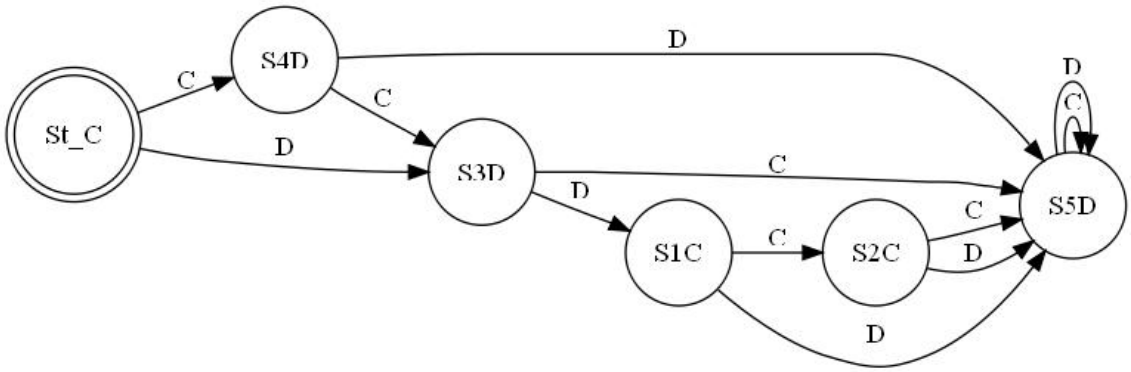


**Figure 3: An agent in the six-turn IPD that waits a single turn before test defecting. $\tau \leq 2$, k = 2.**

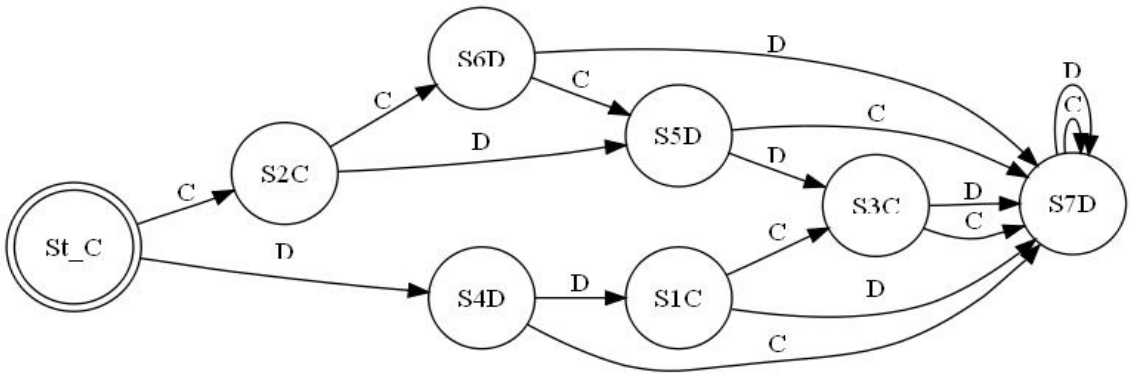


**Figure 4: An agent in the six-turn IPD that waits two turns before test defecting. $\tau \leq 2$, k = 3.**

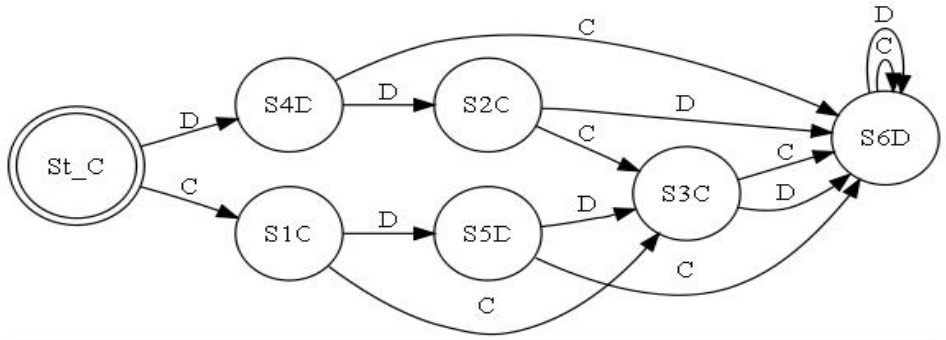


**Figure 5: An agent in the six-turn IPD that waits three turns before defecting. Notice that at this stage, it no longer resumes cooperation after these three turns have passed. $\tau \leq 2$, $\mathbf{k = 4}$.**

## 5. CONCLUSION

In environments with possibilities for cooperation and conflict, the cognitive hierarchy model provides an approach to planning that combines background expectations about other agents with limited strategic reasoning. In this paper, we have used POMDP inference to explore the strategies derived by the cognitive hierarchy model in limited sessions of repeated prisoners' dilemma.

The solutions show how the cognitive hierarchy model operationalizes the intuitive tradeoffs that characterize interactions in these potentially problematic situations. Agents must challenge their opponents to discover any weaknesses, but they must also engage their opponents to earn their trust. The appropriate balance between these goals, as represented by the round when the agent chooses to test the opponent with a defection, varies as a function both of the distribution of different baseline opponent policies in the model and the extent to which opponents themselves do additional rounds of strategic thinking.

The appropriate balance also depends on the actions available to the agent. In some settings, the cognitive hierarchy model is able to plan truthful announcements of future intentions as a credible way of simultaneously challenging opponents and building trust. Communication thereby improves agents' outcomes.

These preliminary simulations offer an encouraging direction for deeper work on communication and strategy in multi-agent interaction. In future work, we aim both for mathematical results, describing the relationship of baseline strategies, nested inference, and communicative action on planning and performance, and for empirical results, particularly focusing on the extent to which the cognitive hierarchy model can model people's expectations and adaptations and thereby derive agent policies that are able to interact more effectively with users.

## Acknowledgments

Thanks to the anonymous reviewers. This research was supported by NSF HSD-0624191.

## 6. REFERENCES

[1] C. F. Camerer. *Behavioral Game Theory*. Princeton, 2003.

[2] C. F. Camerer, T.-H. Ho, and J.-K. Chong. Sophisticated experience-weighted attraction learning and strategic teaching in repeated games. *Journal of Economic Theory*, 104:137–188, 2002.

[3] C. F. Camerer, T.-H. Ho, and J.-K. Chong. A cognitive hierarchy model of games. *Quarterly Journal of Economics*, 119:861–898, 2004.

[4] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Problems*. PhD thesis, Department of Computer Science, Brown University, May 1998.

[5] Y. Gal. *Reasoning about Rationality and Beliefs*. PhD thesis, Harvard, 2006.

[6] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[7] J. M. Keynes. *The General Theory of Employment, Interest, and Money*. 1936.

[8] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan and Claypool, 2008.

[9] Y. Shoham. Computer science and game theory. *Communications of the ACM*, 51(5), 2008.

[10] J. M. Smith. *Evolution and the Theory of Games*. Cambridge, 1982.